

Technical Report #5-32323
Contract Number NAS8-36955
Delivery Order No. 67

Design Controls
for
Large Order Systems

Final Technical Report for the Period
January 10, 1990 through January 9, 1991

by
Dr. George B. Doane III

Research Institute
The University of Alabama in Huntsville
Huntsville, AL 35899

Prepared for :

NASA Marshall Space Flight Center
Huntsville, AL 35812

NASA National Aeronautical and Space Agency		Report Document Page	
1. Report No. 5-32323	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle Design Controls for Large Order Systems		5. Report Due January 09, 1991	
		6. Performing Organization Code Research Institute, UAH	
7. Author(s) George B. Doane III, Mike Jones		8. Performing Organization Report No. Acct. 5-32323	
		10. Work Unit No. Unknown	
9. Performing Organization Name and Address UAH Research Institute RI E-47 Huntsville, AL 35899		11. Contract or Grant No. NAS8-36955, D.O. 67	
		13. Type of report and Period covered Final Technical 10 January 1990 to 09 January 1991	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-001 Marshall Space Flight Center, AL 35812		14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract The output of this task will be a program plan which will delineate how MSFC will support and implement its portion of the Inter-Center Computational Controls Program Plan. Another output will be the results of looking at various multibody/multidegree of freedom computer programs in various environments.			
17. Key Words (Suggested by Authors(s)) Large Scale Computing, Controls for Large Order System, MSFC/NASA Controls Plan.		18. Distribution Statement	
19. Security Class. (of this report) Unclassified	20. Security Class. (of this page) Unclassified	21. No. of pages 7 + Appendix	22. Price TBA

PREFACE

This technical report was prepared by the staff of the Research Institute, The University of Alabama in Huntsville. This report is to serve as documentation of technical work performed under contract number NAS8-36955, Delivery Order 67. Dr. George B. Doane III was Principal Investigator. Mr. Mike Jones provided assistance. Mr. John Sharkey, Structures and Dynamics Laboratory, Science and Engineering, MSFC/NASA, provided technical coordination.

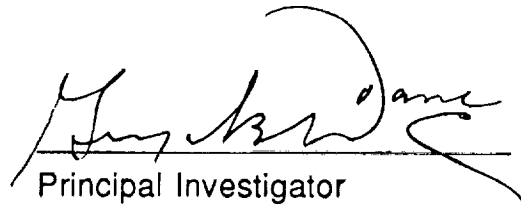
The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official MSFC/NASA position, policy, or decision unless so designated by other official documentation.

Except as otherwise provided by the contract, the distribution of any contract report in any stage of development or completion is prohibited without the approval of the Contracting Officer.

Prepared for

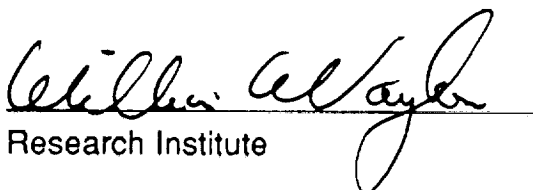
MSFC

Marshall Space Flight Center, AL 35812



Principal Investigator

Approval:



Research Institute

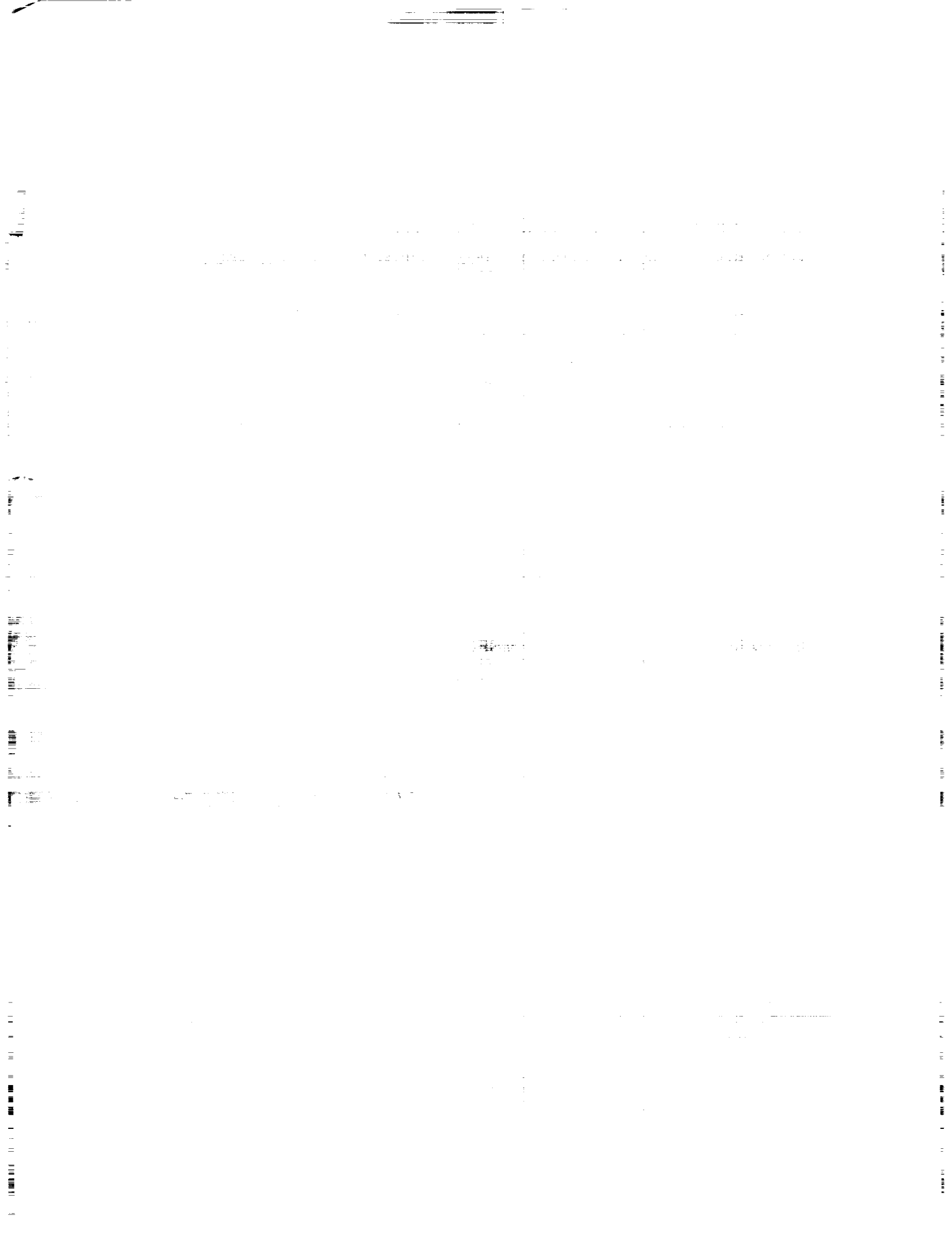


TABLE OF CONTENTS

BACKGROUND	1-1
APPROACH	1-2
RECOMMENDATIONS.....	1-4
CONCLUSIONS.....	1-7
Appendix A: SPOCK.....	A-1
Appendix B: AUTOSIM Code Generator	B-1
Appendix C: AMES/USRA Workstation Workshop	C-1
Appendix D: SD/FAST on the VAX and AD-100.....	D-1
Appendix E: DADS.....	E-1
Appendix F: Center for Simulation and Design.....	F-1
Appendix G: WBS	G-1

BACKGROUND

NASA has recognized that continually advancing computational capability in all its many facets is an absolute necessity to carry out its present and future roles and missions. As a premier research and development agency of the U.S. Government NASA-MSFC is forecasting its needs in the larger context of national policies. It is recognized that MSFC's current and upcoming needs deal in large measure (but not exclusively) with the control and utilization of those elements of the nation's space effort that deal with large propulsion systems, launch vehicles and orbital or payload structures (including the Space Station and its appendages). Although policy at the National level is still evolving, it is safe to assume that constrained resources are all that are foreseeable in the near future. Therefore, careful planning to allot available computational resources is mandatory. This effort addresses itself to questions pertaining to this allotment. Material gathered during the course of this investigation is presented as are specific recommendations of actions that MSFC should consider taking.

APPROACH

To form the basis of recommendations as to how MSFC should allot (and justify) its computational resources, a number of different activities were performed. These included in house (at UAH) computing projects; a trip to AMES Research Center to attend a workstation workshop; various demonstrations held at the UAH Visualization and Simulation laboratory; discussion with principals in the field and literature searches. The details of many of these activities are contained in the appendices.

Specifically, these activities include the following. Discussions with and a presentation by the SDI sponsor and Georgia Institute of Technology (GIT) developer of the Special Purpose Operational Computing Kernel (SPOCK). This device is a unique parallel functional processor being developed specifically for SDI types of interceptor missions. Detail of SPOCK are contained in Appendix A. A discussion was held with Dr. Mike Sayers at the University of Michigan Transportation Institute. Dr. Sayers has developed AUTOSIM, a code generator for systems of rigid bodies. AUTOSIM, is written in common LISP and runs on a MAC computer. It does not have provision for flexible mode incorporation, is copyrighted by the University of Michigan and costs 9000 dollars. Information about it is contained in Appendix B. A fascinating workshop on workstations technology was attended which was put on at AMES mainly by USRA. This workshop covered numerous applications of computers, but in the main to subjects far afield from MSFC's needs and indeed was very futuristic in many of its aims, goals, and concepts. A full accounting of the workshop is enclosed as Appendix C. At MSFC's suggestion, arrangements were made to borrow a limited time demonstration copy of SD/FAST SD/EXACT. Before receiving this program, the author spent a week at ADI gaining familiarity with the ADSIM language. This is a program which generates the equations of motion (in FORTRAN) of multiple connected rigid bodies. In addition, SD/FAST has

the necessary software "hooks" to let it be run on the Applied Dynamics AD-100 which is a parallel processor digital machine optimized to the computational tasks found in simulating continuous systems and capable of 20-MFLOPS. UAH has one of these machines and MSFC has at least four, two of which are fiber optic linked and dedicated to real time SSME simulation. This code was exercised on the VAX front end of the AD-100 to generate equations of motion which were then executed along with control logic programmed in the ADSIM language (Version VII) on the AD-100. Dr. Li USRA/ED-12 also ran a problem of his own. His results were transmitted to MSFC on a 5 1/4" floppy disk for his use there. This was a very interesting exercise and is covered at some length in Appendix D. Once again at MSFC's suggestion, two demonstrations of CADSI's DADS (Dynamic Analysis and Design System) software were held at the UAH Visualization and Simulation (V&S) Laboratory. This software system includes capabilities for static, kinematic, dynamic analysis, inverse dynamics, inclusion of vibratory model information, advanced graphics and a library of feedback, control and hydraulic components. Both demonstrations were attended by MSFC personnel, as well as by some personnel employed by local industry e.g., TBE. Information on DADS is covered in Appendix E. MSFC's TREETOPS program has just been transferred into the V&S lab, but time ran out on exercising it in the AD-100 environment. In addition, of course, considerable general literature was read covering the field.

RECOMMENDATIONS

Based upon the information amassed during the activities outlined above, the following near term recommendations for MSFC activity emerged.

One; even though it did not appear that the AMES/USRA, computer activities are very applicable to MSFC programs at this time, they should be monitored on a continuing basis. It seemed clear that JPL, AMES, Stanford and others have a "good thing going" and have their own ideas, largely science oriented, as to how things should go. They also clearly have the ear of NASA Headquarters.

Two; find the means to join the Center for Simulation and Design Optimization of Mechanical systems (see Appendix F for some detailed information). GSFC and LARC are members along with many other Government and Industrial organizations. This organization is advancing all the aspects of Mechanical Simulation that appear quite applicable to MSFC programs and projects.

Three; start an activity at MSFC using several optically linked AD-100 computers currently available at MSFC modeling an LSS of some complexity. This to be done in real time with provision for using the AD-100's Real-Time/Input Output devices connected to a control computer. This would develop some hands on experience with real-time parallel computing (with HWIL too). Not only would this have minimum initial cost impact, but also if a verified model of say the LSS beams hanging in building 4619 were programmed many more control strategies per unit time could be explored. Because of its possession of an AD-100 the UAH V&S Lab could be a substantial contributor to this activity.

Four; if TREETOPS is to be maintained as a contemporary tool then it should have continued effort on its graphics e.g., add color, perhaps expand the number of computing engines especially workstations on which it can readily be installed and continue algorithmic work to enhance its speed of execution. A somewhat more ambitious avenue would be to re-code TREETOPS for running on massively parallel machines and provide the necessary I/O so that real-time simulation with HWIL can be readily accomplished.

Five; initiate an effort to become integrated into the massively parallel computer world. A current example of this type of effort is the 15 million dollar contract that Intel has been awarded to build a machine with 32 billion floating point operations per second (theoretical) using 528 of Intel's i 860 microprocessors. It is supposed to be completed by next spring for the Concurrent Supercomputing Consortium. This is a group of 14 U.S. research organizations (mostly Federal) led by the California Institute of Technology. At the present time, the so called Delta System is to be applied to military and "grand challenges" (e.g. global warming, chemical reactions). Why not try to make LSS control one of the "grand challenges"?

Six; hold meetings of national interest on the subject of computing and its applications to matters of interest here in Huntsville. UAH's Beville Center is equipped to handle all the arrangements if it is so desired.

Seven; attend several well chosen symposia and national meetings each year on this subject. Present paper(s) if possible.

Eight; establish and maintain meaningful relationships with the computer related parts of regional Universities so that bilateral enrichment of each institution occurs at an

intellectual level and also to assure a supply of properly trained personnel into the workforce.

These recommendations are presented in WBS form in Appendix G.

CONCLUSIONS

A large computing thrust is a national priority. Large scale computing is pervasive in that it effects directly all aspects of national activity. In the physical world, typified by Science and Engineering, its applications are quite clear. Not only does it hold promise for reducing experimentation on costly hardware, but also it holds considerable promise for accomplishing tasks heretofore unattainable.

Any technical organization must guard against being left behind as computer related technology progresses. One of the most fascinating, but very hard to deal with facts is that the half life of given computer systems appears to be on the order of a few months to a little more than a year. It must be carefully understood that advances occur in both hardware and software. Both types of advances are often of a fundamental nature (e.g. massively parallel hardware, very-long-instruction word architecture and new algorithms for real-time computing and both have to be pursued simultaneously to keep abreast of the current state of the art.

MSFC has interests in such things as robotics, LSS, propulsion systems and large structures, all of whom can profit from state of the art computing capability. Indeed, if such capability is not in place at MSFC, its technological viability and indeed its credibility will be in question. In this report, eight recommendation ranging from modest and immediate to far reaching and for the future are presented. These recommendations are offered to stimulate thought and serve as a catalyst to bring about a planned computing oriented thrust at MSFC.

Appendix A

Special Purpose Operational Computing Kernel (SPOCK)

Special thanks are due to Mr. Buster E. Kelley of the U.S. Army Strategic Defense Command. Through his good offices the Georgia Institute of Technology developers (Dr. Alford) came to MSFC and made a presentation on SPOCK.

A new innovative concept, the SPOCK, is being developed by GIT. The SPOCK concept, which utilizes an array of function processors connected by a solid state crossbar matrix, provides three major benefits over conventional processing concepts. These are: 1) One-to-oneness with the block diagram of the system implemented, 2) emulation of flight hardware, and 3) real-time performance evaluation of flight hardware prior to flight hardware fabrication.

Arrangements were made with GIT to travel to Atlanta and exercise the machine on a problem of interest to MSFC, however this never occurred.

Appendix B

AUTOSIM Code Generator

Through Mr. J. Munson of ADI, we were made aware of the AUTOSIM code. This code was developed by Dr. Mike Sayers of the University of Michigan. It deals only with connected rigid bodies (no vibrational modes). In a conversation with Dr. Sayers, it became apparent that the users for whom it was developed were the automotive designers. I was surprised therefore that vibration was not a consideration. For what it will do , 9000 dollars seems pricey (especially when TREETOPS is in the public domain) but it will run on a MAC. Note on page 7 of the enclosure a chart comparing various code efficiencies.

ADI APPLIED DYNAMICS INTERNATIONAL

AUTOSIM™ Code Generator for Systems of Rigid Bodies

J. Munson

Applied Dynamics International

AUTOSIM was developed by Dr. Mike Sayers at the University of Michigan Transportation Research Institute.

It is now available for purchase from the University of Michigan.

The descriptions of AUTOSIM that follow are taken from materials produced by Mike Sayers.

AUTOSIM Technical Summary

AUTOSIM is a computer language that **automatically** generates computationally efficient **simulation** programs for mechanical systems composed of multiple rigid bodies. It formulates the equations of motion symbolically, and then writes a Fortran or ADSIM program to solve them.

How Does AUTOSIM Formulate the Equations of Motion?

The analyses performed automatically by AUTOSIM define a “multibody formalism” that:

- Builds on Kane’s method
- Accommodates any force or moment that can be expressed by the analyst using either equations or external routines
- Accommodates any joint with constant geometric properties
- Accommodates tree topologies and closed kinematical loops
- Accommodates nonholonomic constraints

B-6

How Does AUTOSIM Formulate the Equations of Motion?

The analyses performed automatically by AUTOSIM define a “multibody formalism” that:

- Writes custom numerical algorithms when closed-form solutions are unfeasible, e.g.,
 - Initial values are computed for closed kinematical loops
 - Dynamic constraint violation is eliminated with customized numerical correction
- Includes modeling heuristics (masses may be lumped together, etc.)
- Chooses recursive or non-recursive formulations depending on topology
- Produces explicit (uncoupled) equations for the derivatives of the state variables

How Does AUTOSIM Optimize Code?

- Traditional algebraic and trigonometric simplifications are made (terms that are zero are not included, terms that cancel are dropped, $\sin^2x + \cos^2x \rightarrow 1$, etc).
- Formal treatment is used to drop negligibly small terms in equations.
- Constant expressions are factored out and pre-computed. The efficiency is the same as if numerical values were specified for all constants.
- Intermediate variables are introduced for all repeated expressions to avoid redundant computations.
- The equations of motion are symbolically uncoupled.
- Terms that do not appear in the equations of motion are removed.

B-8

Efficiency Comparisons

Total of Multiply/Divide Operations

Formulation	#1	#2	#3
MACSYMA (Hussain and Noble)			5406
Walker and Orin			1627
MACSYMA (Kane and Nielan)			858
Kane and Levinson			732
SD/EXACT			718
SD/FAST	1094		576
Wampler			448
SYMBA (Nielan)		760	384
AUTOSIM	792	455	353
AUTOSIM with some "small" variables	552		

- #1** Spacecraft #1: 5 bodies (main body, moveable camera and flexible boom), 10 d.o.f.
- #2** Spacecraft #2: 5 bodies (main body, 4 antennae), 10 d.o.f.
- #3** Stanford Arm robot: 6 bodies, 6 d.o.f.

Conclusion

ADI feels that a simulation code generator such as AUTOSIM may be a valuable tool.

If you are interested, please contact Jon Munson at ADI (313) 973-1300.

T1: thruster torque #1: Expression = F(5); Direction = [b1]. Acts on the bus from the inertial reference

T2: thruster torque #2: Expression = F(6); Direction = [b2]. Acts on the bus from the inertial reference

T3: thruster torque #3: Expression = F(7); Direction = [b3]. Acts on the bus from the inertial reference

NO COMMENT

ECTOR C(10), F(7), P(32), S(10), Z(243), Q(10), U(10)

Define default values for parameters

ATA BB = 10.0 ,1 N-m-s, coefficient in term in negative boom-torque Z
BCLOCK = 20.0 ,1 N-m-s, coefficient in term in torque from clock motor
GYRO = 2.0 ,1 ?, coefficient in term in argument to THRUST in coefficient in thruster torque #1
IB11 = 115.0 ,1 kg-m², moment of inertia of B
IB12 = -14.0 ,1 kg-m², product of inertia of B
IB13 = 14.0 ,1 kg-m², product of inertia of B
IB22 = 316.0 ,1 kg-m², moment of inertia of B
IB23 = -34.6 ,1 kg-m², product of inertia of B
IB33 = 440.0 ,1 kg-m², moment of inertia of B
IC = 0.35 ,1 kg-m², moment of inertia of C
ID11 = 4.85 ,1 kg-m², moment of inertia of D
ID12 = 0.41 ,1 kg-m², product of inertia of D
ID13 = -0.07 ,1 kg-m², product of inertia of D
ID22 = 2.2 ,1 kg-m², moment of inertia of D
ID23 = -0.54 ,1 kg-m², product of inertia of D
ID33 = 5.5 ,1 kg-m², moment of inertia of D
IP1 = 27.2 ,1 kg-m², moment of inertia of P
IP2 = 0.2 ,1 kg-m², moment of inertia of P
KB = 2000.0 ,1 N-m/rad, coefficient in term in negative boom-torque Z
KCLOCK = 3500.0 ,1 N-m/rad, coefficient in term in torque from clock motor
L1 = 1.5 ,1 m, negative coordinate of attachment point for the camera in dir 3
L2 = 0.75 ,1 m, negative coordinate of mass center of the clock in dir 3
L3 = 0.1 ,1 m, coordinate of attachment point for the camera in dir 2
L5 = 0.22 ,1 m, negative coordinate of mass center of the camera in dir 2
L6 = 0.2 ,1 m, negative coordinate of mass center of the camera in dir 2
L7 = 1.2 ,1 m, negative coordinate of attachment point for E in dir 2
L8 = 3.3 ,1 m, negative coordinate of mass center of the boom in dir 2
LTT1 = 0.23 ,1 m, coefficient in thruster torque #1
LTT2 = 0.21 ,1 m, coefficient in thruster torque #2
LTT3 = 0.31 ,1 m, coefficient in thruster torque #3
MB = 410.0 ,1 kg, mass of B
MC = 6.8 ,1 kg, mass of C
MD = 57.5 ,1 kg, mass of D
MP = 10.7 ,1 kg, mass of P

ispecs STEPTIME = 0.02 ,1 sec, simulation time step
ENDTIME = 30.0 ,1 sec, simulation stop time

Define initial values for variables

A Q0(1) = 0.0 ,1 m, Translation of B0 relative to O, [n1].
Q0(2) = 0.0 ,1 m, Translation of B0 relative to O, [n2].
Q0(3) = 0.0 ,1 m, Translation of B0 relative to O, [n3].
Q0(4) = 0.0 ,1 rad, Rot. of Bpp relative to N about axis #1.

TLE "small-variable spacecraft"

MENT

is code was generated by AUTOSIM 1.0 B9 on June 1, 1990.

The Regents of The University of Michigan, 1989, 1990. All rights reserved.

he small-variable spacecraft is represented mathematically by 20
rdinary differential equations that describe its kinematical and
ynamical behavior. It is composed of 5 bodies and has 10 degrees of
freedom.

BODIES

is (B); parent=N; 6 coords: Q(1) Q(2) Q(3) Q(4) Q(5) Q(6)
lock (C); parent=B; 1 coord: Q(7)
amera (D); parent=C; 1 coord: Q(8)
parent=B; 1 DOP: Q(9)
om (F); parent=E; 1 coord: Q(10)

MULTIBODY COORDINATES

Q(1): Translation of B0 relative to O, [n1]. (m)
Q(2): Translation of B0 relative to O, [n2]. (m)
Q(3): Translation of B0 relative to O, [n3]. (m)
Q(4): Rot. of Bpp relative to N about axis #1. (rad)
Q(5): Rot. of Bp relative to Bpp about axis #2. (rad)
Q(6): Rot. of B relative to Bp about axis #3. (rad)
Q(7): Rot. of C relative to B about axis #3. (rad)
Q(8): Rot. of D relative to C about axis #1. (rad)
Q(9): Rot. of E relative to B about axis #3. (rad)
Q(10): Rot. of F relative to E about axis #1. (rad)

DEPENDENT SPEEDS

U(1): Abs. trans. speed of BCM, axis 1. (m/s)
U(2): Abs. trans. speed of BCM, axis 2. (m/s)
U(3): Abs. trans. speed of BCM, axis 3. (m/s)
U(4): Abs. rot. speed of B about axis #1. (rad/s)
U(5): Abs. rot. speed of B about axis #2. (rad/s)
U(6): Abs. rot. speed of B about axis #3. (rad/s)
U(7): Rot. speed of C relative to B, axis 3. (rad/s)
U(8): Rot. speed of D relative to C, axis 1. (rad/s)
U(9): Rot. speed of E relative to B, axis 3. (rad/s)
U(10): Rot. speed of F relative to E, axis 1. (rad/s)

MOMENTS

T1: boom-torque Z; Expression = -F(1); Direction = [b3]. Acts on the
boom from the bus

T2: boom-torque X; Expression = -F(2); Direction = [e1]. Acts on the
boom from the bus

LOCKT: torque from clock motor; Expression = F(3); Direction = [b3].
Acts on the clock from the bus

AMT: torque from camera motor; Expression = F(4); Direction = [c1].
Acts on the camera from the clock

Each derivative evaluation requires 544 multiply/divides, 472 add/subtracts, and 8 function/subroutine calls.

Calculate trig functions of state variables

```
S(7) = SIN(Q(7))
S(8) = SIN(Q(8))
C(7) = COS(Q(7))
C(8) = COS(Q(8))
```

Kinematical equations

```
Z(1) = P(1)*U(5)
Z(2) = P(1)*U(4)
Z(3) = -(U(2) -Z(2))
Q'(1) = (Q(5)*U(3) + U(1) + Z(1) + Q(6)*Z(3))
Z(4) = (U(1) + Z(1))
Q'(2) = -(Q(4)*U(3) -U(2) + Z(2) -Q(6)*Z(4))
Q'(3) = -(-U(3) + Q(4)*Z(3) + Q(5)*Z(4))
Q'(4) = -(Q(6)*U(5) -U(4))
Q'(5) = (Q(6)*U(4) + U(5))
Q'(6) = (U(6) -Q(5)*Q'(4))
Q'(7) = U(7)
Q'(8) = U(8)
Q'(9) = U(9)
Q'(10) = U(10)
```

External subroutines and extra variables

```
CLKCMD, CAMCMD = CMD(SYSTEM_TIME)
```

define expression for boom-torque z

```
P(1) = (KB*Q(9) + BB*U(9))
```

define expression for boom-torque x

```
P(2) = (KB*Q(10) + BB*U(10))
```

define expression for torque from clock motor

```
T(3) = (KCLOCK*(CLKCMD -Q(7)) -BCLOCK*U(7))
```

define expression for torque from camera motor

```
T(4) = (KCLOCK*(CAMCMD -Q(8)) -BCLOCK*U(8))
```

define expression for thruster torque #1

```
T(5) = LTT1*THRUST(SYSTEM_TIME, 1, (Q(4) + GYRO*U(4)))
```

define expression for thruster torque #2

```
T(6) = LTT2*THRUST(SYSTEM_TIME, 2, (Q(5) + GYRO*U(5)))
```

define expression for thruster torque #3

```
T(7) = LTT3*THRUST(SYSTEM_TIME, 3, (Q(6) + GYRO*U(6)))
```

ynamical equations

ORIGINAL PAGE IS
OF POOR QUALITY

QE(5) = 0.0 ; rad, Rot. of Bp relative to Bpp about axis #2.
 QE(6) = 0.0 ; rad, Rot. of B relative to Bp about axis #3.
 QE(7) = 0.8584073464102069 ; rad, Rot. of C relative to B about axis #3.
 QE(8) = -0.5 ; rad, Rot. of D relative to C about axis #1.
 QE(9) = 0.0 ; rad, Rot. of E relative to B about axis #3.
 QE(10) = 0.0 ; rad, Rot. of F relative to E about axis #1.
 UE(1) = 0.0 ; m/s, Abs. trans. speed of BCM, axis 1.
 UE(2) = 0.0 ; m/s, Abs. trans. speed of BCM, axis 2.
 UE(3) = 0.0 ; m/s, Abs. trans. speed of BCM, axis 3.
 UE(4) = 0.0 ; rad/s, Abs. rot. speed of B about axis #1.
 UE(5) = 0.0 ; rad/s, Abs. rot. speed of B about axis #2.
 UE(6) = 0.0 ; rad/s, Abs. rot. speed of B about axis #3.
 UE(7) = 0.0 ; rad/s, Rot. speed of C relative to B, axis 3.
 UE(8) = 0.0 ; rad/s, Rot. speed of D relative to C, axis 1.
 UE(9) = 0.0 ; rad/s, Rot. speed of E relative to B, axis 3.
 UE(10) = 0.0 ; rad/s, Rot. speed of F relative to E, axis 1.

REGION initial

PRECOMPUTED CONSTANTS

P(1) = L2*MC/(MB + MC)
 P(2) = (L7 + L8)
 P(3) = (ID11 - ID33)
 P(4) = (ID22 - ID33)
 P(5) = L5*MD
 P(6) = L6*MD
 P(7) = (L7 + L8)*MF
 P(8) = L8*MF
 P(9) = (MB + MC + MD + MF)
 P(10) = (IC + MB*(L2*MC)**2/(MB + MC)**2 + MC*(L2*(1 - MC/(MB + MC)))**2 + IF1 + IB11)
 P(11) = (IC + MB*(L2*MC)**2/(MB + MC)**2 + MC*(L2*(1 - MC/(MB + MC)))**2 + IF2 + IB22)
 P(12) = (MF*(L7 + L8)**2 + IF1 + IB33)
 P(13) = ((L5*L5 + L6*L6)*MD + ID11)
 P(14) = (L8*(L7 + L8)*MF + IF1)
 P(15) = (MF*L8*L8 + IF1)
 P(16) = 1.0/P(15)
 P(17) = P(8)*P(16)
 P(18) = P(8)*P(17)
 P(19) = (P(9) - P(18))
 P(20) = 1.0/P(19)
 P(21) = P(8)*P(20)
 P(22) = P(14)*P(16)
 P(23) = P(8)*P(22)
 P(24) = P(14)*P(17)
 P(25) = P(14)*P(22)
 P(26) = P(1)*P(8)
 P(27) = L8*P(8)
 P(28) = L7*P(8)
 P(29) = (IF1 + P(27))
 P(30) = L7*MF
 P(31) = P(20)*P(23)
 P(32) = (P(12) - P(25))

END REGION

NAMEIC continuous

$$Z(62) = -(-L1*(-Z(5)*Z(6) + Z(40) + Z(41)) + L5*(Z(9)*Z(14) - Z(43)) + L6*(Z(8)*Z(14) + Z(44)))$$

$$Z(63) = ((U(4) + Q(9)*U(5) + 2.0*U(10))*Z(17) - Q(10)*Z(45))$$

$$Z(64) = ID13*Z(8)$$

$$Z(65) = ID12*Z(9)$$

$$Z(66) = ID23*Z(8)$$

$$Z(67) = ID12*Z(14)$$

$$Z(68) = (ID13*Z(39) + ID33*Z(43) + ID23*Z(44) - Z(9)*(-ID22*Z(14) + Z(64) + Z(65)) + Z(14)*(-ID11*Z(9) + Z(66) + Z(67)))$$

$$Z(69) = (ID23*Z(9) + ID13*Z(14))$$

$$Z(70) = (ID12*Z(39) + ID23*Z(43) + ID22*Z(44) + Z(8)*(P(3)*Z(14) + Z(64) + Z(65)) - Z(14)*Z(69))$$

$$Z(71) = (ID11*Z(39) + ID13*Z(43) + ID12*Z(44) - Z(8)*(P(4)*Z(9) + Z(66) + Z(67)) + Z(9)*Z(69))$$

$$Z(72) = MD*Z(25)$$

$$Z(73) = MD*Z(23)$$

$$Z(74) = MD*Z(21)$$

$$Z(75) = ID13*Z(10)$$

$$Z(76) = ID12*Z(11)$$

$$Z(77) = ID11*C(7)$$

$$Z(78) = MD*Z(28)$$

$$Z(79) = MD*Z(27)$$

$$Z(80) = MD*Z(26)$$

$$Z(81) = (-ID23*Z(12) + ID22*Z(13) + ID12*S(7))$$

$$Z(82) = (-ID33*Z(12) + ID23*Z(13) + ID13*S(7))$$

$$Z(83) = ID13*Z(12)$$

$$Z(84) = ID12*Z(13)$$

$$Z(85) = ID11*S(7)$$

$$Z(86) = (-Z(83) + Z(84) + Z(85))$$

$$Z(87) = MD*Z(29)$$

$$Z(88) = (ID33*C(8) + ID23*S(8))$$

$$Z(89) = (ID23*C(8) + ID22*S(8))$$

$$Z(90) = (ID13*C(8) + ID12*S(8))$$

$$Z(91) = MF*Z(32)$$

$$Z(92) = MF*Z(31)$$

$$Z(93) = MF*Z(30)$$

$$Z(94) = MF*Z(34)$$

$$Z(95) = MF*Z(33)$$

$$Z(96) = IF1*Q(10)$$

$$Z(97) = IF1*Q(9)$$

$$Z(98) = P(30)*Q(9)$$

$$Z(99) = IF2*Q(10)$$

$$Z(100) = -MD*(Z(10)*Z(60) - Z(11)*Z(61) + Z(62)*C(7))$$

$$Z(101) = MD*(-Z(12)*Z(60) + Z(13)*Z(61) + Z(62)*S(7))$$

$$Z(102) = MD*(Z(60)*C(8) + Z(61)*S(8))$$

$$Z(103) = (F(5) + IC*(U(5)*Z(5) - Z(36)) - Z(10)*Z(68) + Z(11)*Z(70) - Z(62)*Z(72) - Z(61)*Z(73) + Z(60)*Z(74) - Z(71)*C(7))$$

$$Z(104) = (-Z(11)*Z(73) + Z(10)*Z(74) - Q(9)*Z(91) + Z(93) - Z(72)*C(7))$$

$$Z(105) = (-Z(13)*Z(73) - Z(12)*Z(74) + Z(91) - Q(10)*Z(92) - Z(72)*S(7))$$

$$Z(106) = (Q(10)*Z(91) + Z(92) + Z(74)*C(8) - Z(73)*S(8))$$

$$Z(107) = (-Z(11)*Z(79) - Z(10)*Z(80) + Z(94) - Z(78)*C(7))$$

$$Z(108) = (-Z(13)*Z(79) + Z(12)*Z(80) - Z(93) + Q(9)*Z(94) - Z(78)*S(7))$$

$$Z(109) = (Z(95) + Z(80)*C(8) - Z(79)*S(8))$$

$$Z(110) = (IB12 - IF2*Q(9) - Z(25)*Z(78) + Z(23)*Z(79) + Z(21)*Z(80) - Z(11)*Z(81) + Z(10)*Z(82) + Z(32)*Z(93) - Z(30)*Z(94) + Z(31)*Z(95) + Z(97) + Z(86)*C(7))$$

$$Z(111) = Z(62)*Z(87)$$

$Z(5) = (U(6) + U(7))$
 $Z(6) = (U(4)*C(7) + U(5)*S(7))$
 $Z(7) = (U(5)*C(7) - U(4)*S(7))$
 $Z(8) = (Z(5)*C(8) - Z(7)*S(8))$
 $Z(9) = (Z(7)*C(8) + Z(5)*S(8))$
 $Z(10) = S(7)*S(8)$
 $Z(11) = C(8)*S(7)$
 $Z(12) = C(7)*S(8)$
 $Z(13) = C(7)*C(8)$
 $Z(14) = (U(8) + Z(6))$
 $Z(15) = (U(6) + U(9))$
 $Z(16) = (Q(9)*U(4) - U(5))$
 $Z(17) = (Q(10)*Z(15) - Z(16))$
 $Z(18) = P(1)*S(7)$
 $Z(19) = P(1)*C(7)$
 $Z(20) = (L5 - L3*C(8) + L1*S(8))$
 $Z(21) = (Z(20)*C(7) - Z(19)*S(8))$
 $Z(22) = L3*S(8)$
 $Z(23) = (C(7)*(L6 + Z(22) + L1*C(8)) - Z(19)*C(8))$
 $Z(24) = L1*S(7)$
 $Z(25) = (L5*Z(10) + L6*Z(11) - Z(18) + Z(24))$
 $Z(26) = (Z(20)*S(7) - Z(18)*S(8))$
 $Z(27) = ((-Z(18) + Z(24))*C(8) + (L6 + Z(22))*S(7))$
 $Z(28) = (L5*Z(12) + L6*Z(13) - Z(19) + L1*C(7))$
 $Z(29) = (L3 - L5*C(8) + L6*S(8))$
 $Z(30) = P(1)*Q(9)$
 $Z(31) = (P(2) - P(1)*Q(10))$
 $Z(32) = (P(1) + L7*Q(10))$
 $Z(33) = L8*Q(9)$
 $Z(34) = (P(1) - L8*Q(10))$
 $Z(35) = U(4)*U(7)$
 $Z(36) = U(5)*U(7)$
 $Z(37) = Z(36)*C(7)$
 $Z(38) = Z(35)*S(7)$
 $Z(39) = (Z(37) - Z(38))$
 $Z(40) = Z(35)*C(7)$
 $Z(41) = Z(36)*S(7)$
 $Z(42) = (Z(40) + Z(41))$
 $Z(43) = -(U(8)*Z(9) - Z(42)*S(8))$
 $Z(44) = (U(8)*Z(8) - Z(42)*C(8))$
 $Z(45) = U(4)*U(9)$
 $Z(46) = -(U(5)*U(9) - Q(9)*Z(45))$
 $Z(47) = U(6)*U(4)$
 $Z(48) = U(4)*U(5)$
 $Z(49) = U(5)*U(6)$
 $Z(50) = U(4)*U(4)$
 $Z(51) = (Z(50) + U(6)*U(6))$
 $Z(52) = -(U(6)*U(2) - U(5)*U(3))$
 $Z(53) = -(U(4)*U(3) - U(6)*U(1))$
 $Z(54) = Z(5)*Z(7)$
 $Z(55) = (Z(37) - Z(38) + Z(54))$
 $Z(56) = Z(8)*Z(9)$
 $Z(57) = Z(14)*Z(14)$
 $Z(58) = -L1*(-Z(37) + Z(38) + Z(54))$
 $Z(59) = L3*Z(5)*Z(5)$
 $Z(60) = -(L5*(Z(39) + Z(56)) - L6*(Z(57) + Z(9)*Z(9)) - L3*Z(55)*C(8) + (Z(58) - Z(59))*S(8))$
 $Z(61) = (L6*(Z(39) - Z(56)) + L5*(Z(57) + Z(8)*Z(8)) + (Z(58) - Z(59))*C(8) + L3*Z(55)*S(8))$

$$Z(167) = (Z(117) + Z(115)*Z(161))/Z(166)$$

$$Z(168) = (Z(90) - Z(115)*Z(162))/Z(166)$$

$$Z(169) = (Z(135) + Z(115)*Z(163))/Z(166)$$

$$Z(170) = (Z(133) + Z(115)*Z(164))/Z(166)$$

$$Z(171) = (Z(134) + Z(115)*Z(165))/Z(166)$$

$$Z(172) = P(17)*Z(147)$$

$$Z(173) = P(17)*Z(143)$$

$$Z(174) = (Z(117) + Z(160)*Z(173))$$

$$Z(175) = (P(9) - Z(167)*Z(174))$$

$$Z(176) = (Z(138) - Z(154)*Z(172) - Z(162)*Z(173) + Z(168)*Z(174))/Z(175)$$

$$Z(177) = (Z(118) + P(22)*Z(143) - Z(163)*Z(173) - Z(169)*Z(174))/Z(175)$$

$$Z(178) = (Z(105) + Z(147)*Z(151) - Z(155)*Z(172) - Z(164)*Z(173) - Z(170)*Z(174))/Z(175)$$

$$Z(179) = (Z(108) + Z(143)*Z(159) + Z(156)*Z(172) - Z(165)*Z(173) - Z(171)*Z(174))/Z(175)$$

$$Z(180) = (Z(90) - Z(137)*Z(160))$$

$$Z(181) = (Z(138) - Z(139)*Z(153) - Z(137)*Z(161) + Z(167)*Z(180))$$

$$Z(182) = (P(13) - Z(139)*Z(154) - Z(137)*Z(162) - Z(168)*Z(180) - Z(176)*Z(181))$$

$$Z(183) = (Z(90) + Z(137)*Z(163) - Z(169)*Z(180) + Z(177)*Z(181))/Z(182)$$

$$Z(184) = (Z(140) + Z(139)*Z(155) + Z(137)*Z(164) - Z(170)*Z(180) + Z(178)*Z(181))/Z(182)$$

$$Z(185) = (Z(141) - Z(139)*Z(156) + Z(137)*Z(165) - Z(171)*Z(180) + Z(179)*Z(181))/Z(182)$$

$$Z(186) = (-P(24) + Z(116))$$

$$Z(187) = (Z(135) + Z(160)*Z(186))$$

$$Z(188) = (Z(118) + P(14)*Z(157) - Z(161)*Z(186) - Z(167)*Z(187))$$

$$Z(189) = (Z(90) + Z(162)*Z(186) - Z(168)*Z(187) + Z(176)*Z(188))$$

$$Z(190) = (P(32) + Z(129) + Z(130) + Z(131) - Z(163)*Z(186) - Z(169)*Z(187) - Z(177)*Z(188) - Z(183)*Z(189))$$

$$Z(191) = (Z(123) + P(14)*Z(158) - Z(164)*Z(186) - Z(170)*Z(187) - Z(178)*Z(188) - Z(184)*Z(189))/Z(190)$$

$$Z(192) = (Z(128) - P(14)*Z(159) - Z(165)*Z(186) - Z(171)*Z(187) - Z(179)*Z(188) - Z(185)*Z(189))/Z(190)$$

$$Z(193) = (Z(106) - P(17)*Z(148))$$

$$Z(194) = (Z(104) + P(17)*Z(144))$$

$$Z(195) = (Z(133) + Z(160)*Z(194))$$

$$Z(196) = (Z(105) + Z(148)*Z(150) - Z(153)*Z(193) - Z(161)*Z(194) - Z(167)*Z(195))$$

$$Z(197) = (Z(140) + Z(154)*Z(193) + Z(162)*Z(194) - Z(168)*Z(195) + Z(176)*Z(196))$$

$$Z(198) = (Z(123) + P(22)*Z(144) - Z(163)*Z(194) - Z(169)*Z(195) - Z(177)*Z(196) - Z(183)*Z(197))$$

$$Z(199) = (P(10) + Z(25)*Z(72) + Z(23)*Z(73) + Z(21)*Z(74) + Z(32)*Z(91) + Z(31)*Z(92) - Z(148)*Z(151) - Z(155)*Z(193) - Z(164)*Z(194) - Z(170)*Z(195) - Z(178)*Z(196) - Z(184)*Z(197) - Z(191)*Z(198) + (Z(75) - Z(76) + Z(77))*C(7) + Z(10) * (ID33*Z(10) - ID23*Z(11) + ID13*C(7)) - Z(11)*(ID23*Z(10) - ID22*Z(11) + ID12*C(7)))$$

$$Z(200) = (Z(110) - Z(148)*Z(152) + Z(144)*Z(159) + Z(156)*Z(193) - Z(165)*Z(194) - Z(171)*Z(195) - Z(179)*Z(196) - Z(185)*Z(197) - Z(192)*Z(198))/Z(199)$$

$$Z(201) = (-Z(109) + P(17)*Z(149))$$

$$Z(202) = (Z(107) - P(17)*Z(145))$$

$$Z(203) = (Z(134) + Z(160)*Z(202))$$

$$Z(204) = (Z(108) + Z(145)*Z(157) + Z(153)*Z(201) - Z(161)*Z(202) - Z(167)*Z(203))$$

$Z(112) = Z(68)*C(8)$
 $Z(113) = Z(70)*S(8)$
 $Z(114) = (F(7) + Z(111) - Z(112) - Z(113))$
 $Z(115) = Z(87)*C(7)$
 $Z(116) = (P(7) - Z(115))$
 $Z(117) = Z(87)*S(7)$
 $Z(118) = (-P(7)*Q(9) + Z(98) + Z(117))$
 $Z(119) = Z(25)*Z(87)$
 $Z(120) = Z(10)*Z(88)$
 $Z(121) = Z(11)*Z(89)$
 $Z(122) = Z(90)*C(7)$
 $Z(123) = (IB13 - P(7)*Z(30) + Z(32)*Z(98) - Z(119) + Z(120) - Z(121) + Z(122))$
 $Z(124) = Z(28)*Z(87)$
 $Z(125) = Z(12)*Z(88)$
 $Z(126) = Z(13)*Z(89)$
 $Z(127) = Z(90)*S(7)$
 $Z(128) = (IB23 + P(7)*Z(34) - Z(96) + Z(99) + Z(124) - Z(125) + Z(126) + Z(127))$
 $Z(129) = Z(29)*Z(87)$
 $Z(130) = Z(88)*C(8)$
 $Z(131) = Z(89)*S(8)$
 $Z(132) = (F(3) + Z(111) - Z(112) - Z(113))$
 $Z(133) = (-Z(119) + Z(120) - Z(121) + Z(122))$
 $Z(134) = (Z(124) - Z(125) + Z(126) + Z(127))$
 $Z(135) = (Z(129) + Z(130) + Z(131))$
 $Z(136) = (F(4) + P(5)*Z(60) - P(6)*Z(61) - Z(71))$
 $Z(137) = (P(5)*Z(10) + P(6)*Z(11))$
 $Z(138) = (P(5)*Z(12) + P(6)*Z(13))$
 $Z(139) = -(P(5)*C(8) - P(6)*S(8))$
 $Z(140) = (P(5)*Z(21) + P(6)*Z(23) + Z(75) - Z(76) + Z(77))$
 $Z(141) = (P(5)*Z(26) + P(6)*Z(27) - Z(83) + Z(84) + Z(85))$
 $Z(142) = -(-F(1) - P(26)*Z(47) + P(28)*Z(48) - P(8)*(Z(52) + Q(9)*Z(53)) + P(29)*Z(63))$
 $Z(143) = P(8)*Q(9)$
 $Z(144) = (P(8)*Z(30) + Q(9)*Z(99))$
 $Z(145) = (P(8)*Z(34) - Z(96) + Z(99))$
 $Z(146) = (F(2) + P(8)*(U(5)*U(1) - U(4)*U(2)) + P(27)*((Z(15) + Q(10)*Z(16))*Z(17) - Z(46)) - IF1*Z(46) + (P(28) + P(26)*Q(10))*Z(49) + P(26)*(Z(50) + U(5)*U(5)))$
 $Z(147) = P(8)*Q(10)$
 $Z(148) = (IF1 + P(8)*Z(31))$
 $Z(149) = (P(8)*Z(33) + Z(97))$
 $Z(150) = P(16)*Z(147)$
 $Z(151) = P(16)*Z(148)$
 $Z(152) = P(16)*Z(149)$
 $Z(153) = P(21)*Z(150)$
 $Z(154) = P(20)*Z(139)$
 $Z(155) = (P(20)*Z(106) - P(21)*Z(151))$
 $Z(156) = (-P(20)*Z(109) + P(21)*Z(152))$
 $Z(157) = P(16)*Z(143)$
 $Z(158) = P(16)*Z(144)$
 $Z(159) = P(16)*Z(145)$
 $Z(160) = P(20)*Z(115)$
 $Z(161) = P(21)*Z(157)$
 $Z(162) = P(20)*Z(137)$
 $Z(163) = (-P(31) + P(20)*Z(116))$
 $Z(164) = (P(20)*Z(104) + P(21)*Z(158))$
 $Z(165) = (P(20)*Z(107) - P(21)*Z(159))$
 $Z(166) = (Z(135) - Z(115)*Z(160))$

```

U'(5) = Z(236)
U'(4) = Z(237)
U'(6) = Z(238)
U'(8) = Z(239)
U'(2) = Z(240)
U'(7) = Z(241)
U'(1) = Z(242)
U'(9) = -(P(22)*Z(238) - P(16)*(-Z(142) - Z(145)*Z(236) +
Z(144)*Z(237) - Z(143)*Z(240)) + P(17)*Z(242))
U'(3) = Z(243)
U'(10) = -(P(16)*(Z(146) + Z(149)*Z(236) + Z(148)*Z(237) +
Z(147)*Z(240)) - P(17)*Z(243))

```

END DYNAMIC

EXECUTE 'CONTINUOUS'

**** Code above this line generated entirely by AUTOSIM ****

FUNCTION cmd (clkcmd, camcmd = t)

```

IF t .lt. 1 THEN
  clkcmd = 4 - .PI.
  camcmd = -0.5
ELSEIF t .lt. 11 THEN
  clkcmd = 4 - 0.025 * (t - 1) - .PI.
  camcmd = -0.5 + 0.01 * (t - 1)
ELSE
  clkcmd = 3.75 - .PI.
  camcmd = -0.4
ENDIF

```

END FUNCTION

FUNCTION thrust (thr = t, axis, terror)

VECTOR fire(3), toff(3)

ATA dband = 0.0025

ATA tmin = 0.02

ATA fire = (0,0,0), toff = (0,0,0)

```

IF terror .lt. -dband THEN
  fire(axis) = 1
  toff(axis) = t + tmin
ELSEIF terror .gt. dband THEN
  fire(axis) = -1
  toff(axis) = t + tmin
ELSEIF t .ge. toff(axis) THEN
  fire(axis) = 0
ENDIF

```

thr = fire(axis)

END FUNCTION

ORIGINAL PAGE IS
OF POOR QUALITY

$$Z(205) = (Z(141) - Z(154)*Z(201) + Z(162)*Z(202) - Z(168)*Z(203) + Z(176)*Z(204))$$

$$Z(206) = (Z(128) - P(22)*Z(145) - Z(163)*Z(202) - Z(169)*Z(203) - Z(177)*Z(204) - Z(183)*Z(205))$$

$$Z(207) = (Z(110) - Z(149)*Z(151) + Z(145)*Z(158) + Z(155)*Z(201) - Z(164)*Z(202) - Z(170)*Z(203) - Z(178)*Z(204) - Z(184)*Z(205) - Z(191)*Z(206))$$

$$Z(208) = P(17)*Z(146)$$

$$Z(209) = (Z(102) + Z(208))$$

$$Z(210) = P(17)*Z(142)$$

$$Z(211) = (Z(100) + Z(210))$$

$$Z(212) = Z(160)*Z(211)$$

$$Z(213) = (Z(132) + Z(212))$$

$$Z(214) = Z(153)*Z(209)$$

$$Z(215) = Z(161)*Z(211)$$

$$Z(216) = Z(167)*Z(213)$$

$$Z(217) = (-Z(101) + Z(214) + Z(215) + Z(216))$$

$$Z(218) = Z(154)*Z(209)$$

$$Z(219) = Z(162)*Z(211)$$

$$Z(220) = Z(168)*Z(213)$$

$$Z(221) = Z(176)*Z(217)$$

$$Z(222) = (Z(136) + Z(218) + Z(219) - Z(220) - Z(221))$$

$$Z(223) = P(22)*Z(142)$$

$$Z(224) = Z(163)*Z(211)$$

$$Z(225) = Z(169)*Z(213)$$

$$Z(226) = Z(177)*Z(217)$$

$$Z(227) = Z(183)*Z(222)$$

$$Z(228) = (Z(114) + Z(223) - Z(224) - Z(225) + Z(226) - Z(227))$$

$$Z(229) = Z(146)*Z(151)$$

$$Z(230) = Z(155)*Z(209)$$

$$Z(231) = Z(164)*Z(211)$$

$$Z(232) = Z(170)*Z(213)$$

$$Z(233) = Z(178)*Z(217)$$

$$Z(234) = Z(184)*Z(222)$$

$$Z(235) = Z(191)*Z(228)$$

$$Z(236) = (P(6) - C*(U(4)*Z(5) - Z(35)) + Z(12)*Z(68) - Z(13)*Z(70) + Z(62)*Z(78) - Z(61)*Z(79) + Z(60)*Z(80) + Z(142)*Z(159) + Z(156)*Z(209) - Z(165)*Z(211) - Z(171)*Z(213) + Z(179)*Z(217) - Z(185)*Z(222) - Z(192)*Z(228) - Z(200)*(Z(103) + Z(229) - Z(230) - Z(231) - Z(232) + Z(233) - Z(234) - Z(235)) - Z(71)*S(7))/(P(11) + Z(28)*Z(78) + Z(27)*Z(79) + Z(26)*Z(80) + Z(13)*Z(81) - Z(12)*Z(82) + Z(34)*Z(94) - Z(145)*Z(159) - Z(156)*Z(201) - Z(165)*Z(202) - Z(171)*Z(203) - Z(179)*Z(204) - Z(185)*Z(205) - Z(192)*Z(206) - Z(200)*Z(207) + Z(86)*S(7))$$

$$Z(237) = (Z(103) + Z(229) - Z(230) - Z(231) - Z(232) + Z(233) - Z(234) - Z(235) - Z(207)*Z(236))/Z(199)$$

$$Z(238) = (Z(114) + Z(223) - Z(224) - Z(225) + Z(226) - Z(227) - Z(206)*Z(236) - Z(198)*Z(237))/Z(190)$$

$$Z(239) = (Z(136) + Z(218) + Z(219) - Z(220) - Z(221) - Z(205)*Z(236) - Z(197)*Z(237) - Z(189)*Z(238))/Z(182)$$

$$Z(240) = (-Z(101) + Z(214) + Z(215) + Z(216) + Z(204)*Z(236) + Z(196)*Z(237) + Z(188)*Z(238) - Z(181)*Z(239))/Z(175)$$

$$Z(241) = (Z(132) + Z(212) - Z(203)*Z(236) - Z(195)*Z(237) - Z(187)*Z(238) - Z(180)*Z(239) + Z(174)*Z(240))/Z(166)$$

$$Z(242) = -P(20)*(-Z(100) - Z(210) + Z(202)*Z(236) + Z(194)*Z(237) + Z(186)*Z(238) - Z(137)*Z(239) - Z(173)*Z(240) - Z(115)*Z(241))$$

$$Z(243) = -P(20)*(Z(102) + Z(208) + Z(201)*Z(236) - Z(193)*Z(237) + Z(139)*Z(239) + Z(172)*Z(240))$$

Appendix C

AMES/USRA Workstation Workshop

The "Workshop on NASA Workstation Technology" held in Palo Alto, California, during the week of March 12, 1990 was attended. This work is sponsored by NASA Code R (Lee Holcomb). It is centered in the Research Institute for Advanced Computational Science (RIACS). This is a NASA funded organization formed by USRA "to provide leadership in developing computers" and is headquartered at NASA AMES Research Center. The Workshop was emceed by Dr. Bob Brown, RIACS' technical director, acting under the overall direction of RIACS' assistant director, Dr. Berry Leiner (Dr. Brown's boss). The meetings were actually held at the Hyatt Rickeys. The meeting room was certainly adequate, the noon meals, served buffet styles in an adjoining dining room were outstanding as were the break snacks and the "heavy hors d' oeuvres" served on the second evening during the vendor show.

The Workshop, while exceedingly interesting from the viewpoint of being exposed to new vistas of thought, was in no wise along the lines we had discussed before my going. And at no time did anyone define a Workstation. I will offer my own definition for discussion later. Even at a distance of two weeks it is still not entirely clear what eclecticism was used to choose the variegated program. I will include a brief summary of my impressions of most of the talks but perhaps a synopsis of the keynote address will provide a general "sense of the meeting."

The keynote address was given by Dr. William Bricken, Chief Scientist, Human Interface Technology Center, University of Washington. In his talk Dr. Bricken presented a "Vision of Virtual Reality." In his address he suggested the use of the computer (and an appropriate human sensory environment) as a replacement for reality. One would, in this scheme, seem to be "wearing a computer." In this scenario anything imaginable would be sensorally possible, e.g., the known laws of physics would be irrelevant. It would be possible as an example to sense being on Mars with a population of penguins moving mountains with the flick of a (sensed) finger. A pale subset of this direction of sensing is seen in the very realistic scene generation, flying maneuvers etc currently available in USAF training commands. Of course, he pointed out that carrying this work forward will require new paradigms of computing and boundary mathematics. Some of the audiences' comments and discussions at the break regarding this address included a few catch phrases, e.g., post STARSHIP, sanity, realism, "electronic crack." My own reaction is that with time much of this could be possible and we will then pass through

another door into yet another Brave New World. Stopping short of the ultimate there seems to be considerable use for this sort of thing in education, training, entertainment, and repair work.

Mr. Lee Holcomb gave the charge to the group. A group of us also talked to him during one of the breaks. The subject during the break was the upcoming national High Performance Computing Initiative. Mr. Holcomb had attended a meeting not long ago chaired by the President's science advisor (Dr. Bromley) and attended by OMB, NSF, NASA, and DOD representatives. The information was that the advisor had been roughly handled in Congress because of the lack of an Administration's High Performance Computing Initiative plan and the subsequent lack of any budget for it. This will be rectified in the next ('92) budget and this was agreed to by the OMB representative present. In his charge to the Workshop Mr. Holcomb listed three applications of advanced computing. The first was its use in operations. This included, as examples, both space and aeronautical operations to include real time control (telescience), anomaly identification and correction and scheduling and planning of operations.

His second application was that of science analysis. There are almost unimaginably large volumes of data currently available. These data are being added to daily and as projects such as the Space Station come on line an even more rapid stream of data will occur to fill the data banks. The challenges then include finding ways to allow appropriate people access to these huge data banks, how to couple various disciplines (e.g. plankton biology and oceanography) to produce coupled phenomenological models and just in general how to facilitate the collaborations of multidisciplinary scientists on a global scale. His third and last charge (and as far as I could see the only explicit mention of the totality suggested) was to address how to build complex aerospace systems. He mentioned that computational fluid dynamics, structural design etc are being pursued but that the overall challenge is to pull together or model the interaction of fluid flow, aerodynamics, guidance and control, structures, stealth and so forth so as to produce an integrated, optimal design - or, in the case of something like NASP, perhaps any viable design at all. He concluded expressing hope that the Workshop will encourage an exchange of general information, technology, applications, capabilities vendors and software. He also asked rhetorically what standards should

NASA support? For instance networking and the UNIX versus X-WINDOWS versus...discussion. He also expressed concern over high maintenance cost e.g. 25% of DEC's income is from maintenance work.

There is supposed to be a mailing of hard copy of the presentation materials. A video camera was run during the presentations but as we left we were told that the quality of the video was very poor and only the audio was useable. In all events the media presentation materials were generally either in viewgraph or VCR form. Therefore, I shall set forth below some of my impressions of the various speakers' messages.

Mr. Jay Costenbader of GSFC, mission operations, picked out three workstations usages at GSFC. These were in mission operations, data capture and guidance and control. My impression is that although there are many automation aids via the computers involved there is much human intervention and decision making. He indicated that this is because of the very conservative approach taken by the various system (e.g. Hubble ST) managers. They have looked at the ADA language and found it too restrictive, they (as are a number of others) are moving on to C++.

Mr. Jim Jeletic of GSFC, Flight Dynamics Division, reinforced a number of Mr. Costenbader's points and added a few. To wit, they have to support long term ongoing programs with current (i.e. may be very old) inflight protocols (may go on for years). This makes for a mix of operational systems to support. He sees more ethernet "bridges" and more distributed computing in the future. He pointed out that there is an ADA champion at GSFC thus ADA is used whenever possible. He sees concerns including resource consumption (memory, performance), system compatibility, system complexity (which workstations to use, what operating system) and distributed systems making for harder troubleshooting.

Mr. Mike Wiskerchen of Stanford University had a most interesting story to tell. Under Code R support he undertook to study the application of MIS to NASA operational needs. The vehicle for this study was the process of making sure the Orbiter tiles were ready for flight. Apparently he had worked for NASA at one time. Anyway, he went to

KSC and found that for the tiles only 360,000 sets of initials or signatures are required to release an Orbiter for flight. He found an almost absolute lack of MIS use in the system and as he progressed through the system (he actually performed himself all the jobs in the chain, it took a number of months) he developed some interesting statistics. These included his 85-12-3 rule. This rule is that 85% of the people favor the status quo, 12% actively oppose change, and the 3% support change in the modus operandi. He found that 90% of the people at KSC do not use computers. Thus, technology is 10% of the problems and people 90% of the problem. Only the direct involvement of top management (who do not know about computers now) and the establishment of rewards for technology change (such rewards do not now exist) will cause a change. On the technical side there is a problem. The technology of MIS changes on a 6 to 9 month lifecycle. Programs evolve over 10-15 years . Can you design to replace technology every year or so?

Mr. Randy Davis of the University of Colorado which apparently has a mission operations center for some scientific missions gave a presentation on OASIS. OASIS is their software package (Operations and Science Instrument Support Teleoperations support software). Overall they propose moving control of payloads to the PI's own home premises (versus current gather-everyone-at-a-NASA-center). Their software is written in ADA and used on DEC/VAX, Sun III's, (S IV upcoming). The general theme was to work on a coherent architecture for payload control.

Mr. Contenbader of GSFC returned and talked about TAE + (Transportable Application Environment Plus). This environment separates the users interface from applications programmers. It now has spread, after a slow start, to 500 users and is available from COSMIC. Some technical requirements were given.

Mr. Andrew Potter of MSFC (I cannot find him in the October 1989 MSFC phone book) gave a talk about the goals to be obtained in designing a workstation for the Space Station. It emphasized instruments on a screen ('a la the "glass cockpit") and talked about search strategies. His presentations was not on MSFC viewgraphs.

Mr. Tim Castellano of AMES RC talked about the workstation aspects of the SHOOT project. This project is supposed to fly in the fall of 1992 on STS61. It is a superfluid helium transfer flight demonstration. It includes doing some control and observations of the experiment on MAC IIs and a laptop from the POCC at GSFC. It is an expert based system and has 100 to 125 rules so far.

Ms. Patricia Liggett of JPL aggressively headed a trio of presenters detailing the VANESSA project. It was a system rather hurriedly put together from existing equipment for the Voyager Neptune Encounter Science Support Activity. In a nutshell it allowed the delivery of both engineering and encounter science data in near real time. Of course the Voyager data stream bit rate is quite low so equipment and software speed was not a problem. The problem was to fan out the data from the data stream (not the data base, once in there retrieval is arcane and slow) to a multiplicity of user data formats. JPL furnished some of the on site (at JPL) workstations other PI'S brought their own. This was a one time effort but showed what type of data display etc will be required in the future.

Mr. Graius Martin of JPL's AI group (including Cal Tech's Doyle) described the SHARP system development status. SHARP stands for Spacecraft Health Automated Reasoning Prototype. The objective of this development is to shrink the "army on the ground" which is currently seen to be proliferating with the ever increasing number of spacecraft borne experiments.

The goals include an 80% staff reduction, increased safety because of more data being under surveillance, more reliability because of system wide monitoring and vastly increased productivity because one operator can be spread across a number of flight systems through the use of AI. They have put together a Symbolics based workstation and later port to a Sun Workstation to support the Magellan mission spacecraft. Various aspects are being checked out with the archived Voyager data.

Mr. Tom Engler of MSFC (can not find him in the phone book either) gave an overview of what MSFC hopes to do in automating the HOSC. HOSC is currently actually four centers. One each supports Shuttle, HST, Spacelab (POCC) and Space Station. The

goal is to multiplex people, use emerging technologies, use AI and possibly neural networks etc. Very preliminary design or conceptual goal type of paper. Questions to him were mainly had he looked at the other existing control centers. The answer was yes, at GSFC, JPL, and JSC.

Mr. Jerry Blackburn of JSC covered their work toward updating the shuttle through the development of MPAC (Multipurpose Applications Console). This would represent an update to 386 technology, use ADA and a 1553B bus. They are also developing Space Station DMS and have MacDac and IBM "on the team."

Mr. Eric Hibbard of AMES RC reported on the letting of 4 seven year contracts to DEC, Silicon Graphics, Sun, and Convex concerning workstations and midrange computers. They are to develop graphics standards in software and hardware. They are looking toward animated stereographics, distributed graphics (i.e. created from various non co-located sites) and various rendering tools. They now are using X-windows system (release 4) and advocate CGI extensions.

Mr. Val Watson of AMES RC had a number of recommendations for improvements in Scientific Workstations. See his handout, a copy is enclosed.

Mr. Keith Lane "nationally well known for multimedia" who just left Olivetti Computer (80% force reduction) talked on a portable work environment (which is not just a portable computer). The gist is that your portable computer is a window to your resources (which may be anywhere). He suggested audio at the 768 Kbyte to 2 M Byte level, video at 90-240 or 10-60 (compressed) and data up to video rates. It seemed to me that all this was very much in the hand wavy hobby shop stage.

Mr. Barry Leiner, Assistant Director of RIACS, described the National Collaboratory (Collaborative + Laboratory). Basically this idea grew from a conference held at Rockefeller University in 1989. A report of that conference is available. It is basically another attempt to deal with world wide knowledge/data explosion. Its tools would

include a digital library (Knowbots, knowledge robots, to include digitizing the world's knowledge), interoperable data, information fusion, smart/remote instruments, joint authority tools and symbolic assistants (the theorist's workbench e.g. mathematical objects versus numbers). NSF has a new initiative in this direction; 3 AOs are for this sort of thing.

Mr. Joe Hale of MSFC (EC23) talked about the development surrounding the use of the Private Eye. This is a static data display instrument which is worn over (or slightly under) one eye and produces a virtual image focused from 10 inches to infinity. It may or is being combined with a Dataglove to provide Space Station documentation which goes along with an operator (versus using a "floating in zero -G" manual). While in early state of development seems promising to me.

Mr. Michael McGreevy of AMES RC put on a razzle dazzle picture show of various geological data. His conclusion was that desktop data presentations are not adequate. Something like the head mounted virtual reality techniques are needed (he buttressed this thesis by showing mosaics from the Moon and Mars).

Because no one else offered their notion of what constitutes or defines a workstation I will, in the spirit of polemics, offer one. I propose to define one as a single person communication node that has access to outside information resources in addition to its own capabilities and storage means.

My conclusion as far as this NASA effort is concerned is that it is a "going concern" what with RIACS off and running. AMES clearly has a large interest and influence in the effort e.g. RIACS is housed there. There is a very strong science flavor to the effort. This science flavor manifests itself in several ways. One way is in the pure development of the various leading edge technologies involved e.g. A.I. Another is in the research and application interests of the present group. This includes planetary exploration, the processing and display of very large scientific data bases e.g. the portrayal of 3D turbulent flow fields created by CFD. The other type of use growing out of the programs (e.g. Space Station) is in operations. Examples include mission operations (reduce the

Army on the Ground) and training about and repair of orbital hardware. The direct application of all this to current known ED-12 programs is not obvious. In the long run faster running generally more capable workstation technology will benefit LSS control, of course. Based on this conference little or no direct hardware in the loop control problems are being addressed specifically (at least of a type applicable to say CASES). Probably monitoring progress reports would be worth the effort especially in real time data collection disciplines. To try and do any serious steering to the direction of the effort would be difficult and require dealing with the entrenched interests of AMES, JPL, and RIACS.

Workshop on NASA Workstation Technology

Agenda

Day One - March 13, 1990

- 8:30 AM** **Registration desk opens.**
Coffee and pastries served in the Ballroom lobby.
- 9:00 AM** **Welcoming and opening remarks.**
Barry Leiner, Peter Denning, and Robert Brown
- 9:15 AM** **Charge and goals for the workshop.**
Lee Holcomb
- 9:30 AM** **Keynote speaker.**
William Bricken, Chief Scientist, Human Interface Technology Center, University of Washington
- 10:10 AM** **Mission Operations Workstations.**
Jay Costenbader, GSFC, *Three Mission Operations Workstation Projects*
Jim Jeletic, GSFC, *Workstation Technology Used for Flight Dynamics Mission Support*
Mike Wiskerchen, Stanford University, *Application of Advanced Workstation Technology to Shuttle Operations*
- 12:00 PM** **Break for lunch.**
Location: *Ballroom C*
- 1:00 PM** **Application Development Environments**
Eric Hardy, CMU, *The SEI User Interface Project*
Randy Davis, University of Colorado, *Oasis: Present and Future*
Jay Costenbader, GFSC, *TAE+*
- 3:00 PM** **Coffee break.**
- 3:15 PM** **Andrew Potter, MSFC, *The SoftPanel Prototype***
Mission Science Workstations.
Tim Castellano, ARC/RIA, *SHOOT*
Patricia Liggett, JPL, *The VNESSA System*
- 5:00 PM** **Technical sessions end.**
- 6:30 PM** **Vendor show, poster and demonstration session.**
Hors d'oeuvres and no-host bar
Ballroom C & D
- 9:30 PM** **Day one ends.**

Day Two - March 14, 1990

- 8:00 AM** **Coffee and pastries.**
Ballroom lobby
- 8:30 AM** **Mission Operations (continued)**
Gaius Martin, JPL, *Sharp*
Tom Engler, MSFC, *MSFC Workstation Lab:SSF and AXF*
Gregory Blackburn, JSC, *SSF Multipurpose Applications Console (MPAC)*
- 10:30 AM** **Coffee break.**
- 10:45 PM** **Science Data Visualization.**
Eric Hibbard, ARC, *Code RCD Graphics*
Leo Blume, JPL, *Linkwinds -- A Prototype Scientific Visualiztation System*
- 12:00 PM** **Break for lunch.**
Location: *Ballroom A*
- 1:00 PM** Val Watson, ARC, *Workstation Applications Office Projects*
Stephen Coles, JPL, *EASE: An Engineering Analysis Subsystem Environment for Real-Time Spacecraft Control*

Productivity and Collaboration Tools.
Keith Lantz, Consultant, *Multimedia Workstations for Collaboration*
- 3:00 PM** **Coffee break.**
- 3:15 PM** Barry Leiner, RIACS, *The National Collaboratory*

Other Workstation Projects.
Larry Foster, Boeing, *TMIS, Micro/host/Integration*
Michael McGreevy, ARC, *Ames Virtual Reality Projects*
- 5:00 PM** **Workshop wrap-up.**
Robert Brown
- 5:15 PM** **Break for the day.**

Day Three - March 15, 1990

The third day of the workshop consists of working sessions at NASA Ames Research Center. We are interested in having 15-25 people work in a small number of groups to capture and document the important parts of the workshop and to begin the process of creating the workshop report.

Key elements of the report could be:

The state of workstation technology inside NASA.

The priorities for future activities.

A structure will be created in which the third-day participants will work. This structure will be reviewed in a plenary session at the beginning of the day. Then, small groups will form and work on a detailed outline of the report.

Who should participate? Speakers, managers, those with opinions or an interest in impacting the report. These people will become reviewers for the report as it is being integrated and edited over the course of the following two months.

- 9:00 AM** **Description of the report-writing process.**
Robert Brown
- 10:00 AM** **Break into groups by session.**
- 12:00 PM** **Break for lunch.**
- 2:00 PM** **Plenary session for presentation of results.**
- 5:00 PM** **Workshop ends.**

Support Personnel

- Bob Brown, Program, Demonstrations
- Dee Doyle, General Scheduling & Management
- Lorraine Fisher, Local arrangements, messages
- Others (Green Badges)

Posterboard Session

- Posters hold 9 (portrait) or 6 (landscape) letter-sized pages
- Give materials to Lauri Howard or Bob Brown
- Posters will be presented at the vendor show

Demonstrations

- Limited demonstration capability
- Sun 3/160 color & monochrome
- SGI IRIS (personal & 4D/220)
- NeXT
- Macintosh (MacOS, MacX)
- IBM RS6000
- Sun SPARCStation & Sun 4/330
- DEC DECStation 3100

Vendor Session

- 6:30-9:30PM, adjoining ballroom
- Hors d'oeuvres served in lobby
- No-host bar
- Demonstrations mostly on vendor equipment
- Video tapes running in the corners

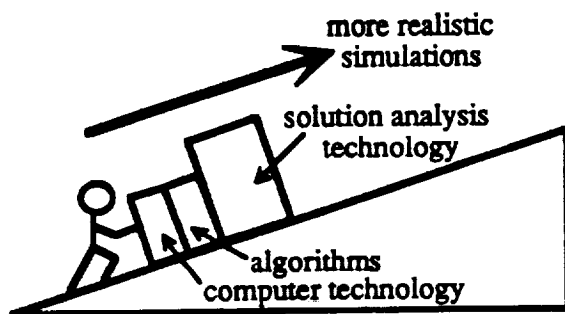
Recommendations for Improvements in Scientific Workstations

Val Watson
NASA Ames Research Center

Outline

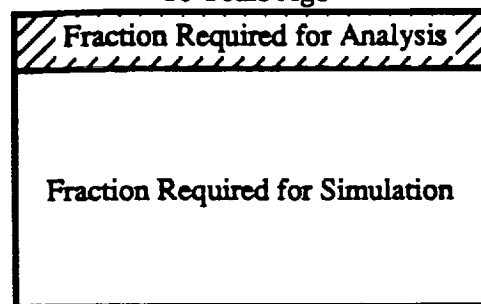
Workstation features important for scientific applications
Illustrations of scientific analysis using visualization
Current capabilities for analysis
Comparison of current capabilities with "ideal"
Recommendations for the future

Limitations to Progress in Computer Simulations

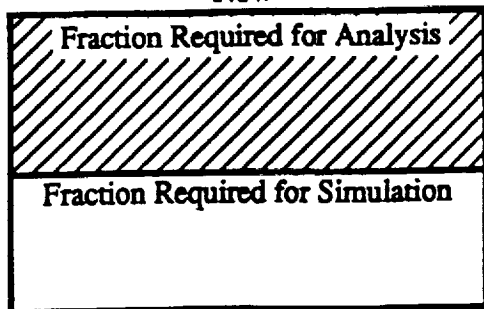


Fraction of Computing Required for Analysis

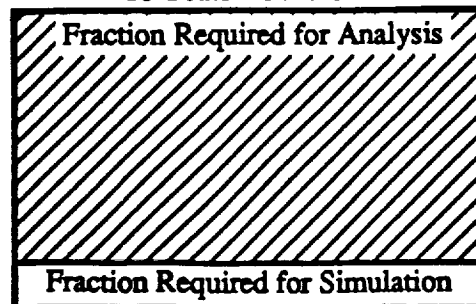
10 Years Ago



Fraction of Computing Required for Analysis Now



Fraction of Computing Required for Analysis 10 Years from Now



Requirements for Rendering the Representations

All currently popular scenes can be created with

- 3D surfaces
- 3D lines
- points

Surfaces composed of 4 sided non-planar polygons
(flat shading sometimes preferred)

Typical scene represented with 10,000 of these
10 "frames/sec" needed to understand dynamics

Therefore, the derived requirement is

100,000 polygons/sec with hidden surfaces removed

Current NAS Workstation Procurement

300 workstations over 3 year period

Graphics performance is most important factor

Threshold of pain for cost approximately \$100K

Most weight given to graphics performance

Performance measured with our benchmarks

- surface and line graphics benchmark
- fluid dynamic code floating point benchmark

NAS Graphics Benchmark

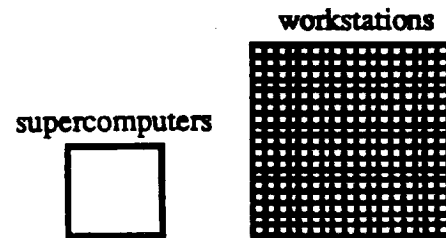
consists of 3D scene of Shuttle with particle traces
and function mapped surface

measures 3D coordinate transformation rate

measures polygon and line rendering rate

- with hidden surface removed
- with mostly 4 sided non-coplanar polygons
- with and without smooth shading
- with and without transparency
- with and without antialiasing

Distribution of Computing Power after the Procurement



Current Work on Software

Combination of old modules into a single program

Changes to take advantage of new workstations

Changes to make visualization more interactive

Visualization Software

Simulation	Visualization and Recording			
	Scene Creation	Scene Viewing	Animation Sequence Creation	Recording on Film and Video
	← PLOT3D →			
	← SURF →			
	← RIP →			
			GAS	
	← FAST →			

Additional Hardware Improvements Recommended

An increase in the field of view of the display to more nearly match the eye's field of view

Effective six-degree of freedom controls

Sound

Voice recognition

- discrete word or phrase now
- continuous voice later

Specialized hardware to support the tasks described in the next viewgraph

Additional Software Improvements Recommended

Software for new data transformations, filters and comparators that will be useful in fluid dynamics

Software to extract the essence of the fluid flows

Software to clearly display the essence of the fluid flows (to create displays highly tuned to the human's cognitive capabilities)

Software to generate pictures for monitoring accuracy and stability of the computer simulations using the human's pattern recognition capabilities

Current Visualization Research

Scott Fisher's "Virtual Environment" project

- Helmet display fills the field of view
- Display corresponds to head position
- Voice recognition
- Data gloves for manual control of objects

(scientist feels like he is inside the simulated field)

Professor Hesselink's research project

- feature extraction using expert system (based on flow field topology rules)
- display that highlights critical features and suppresses "visual clutter"

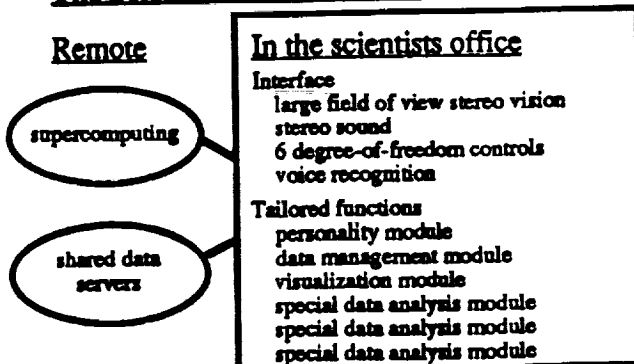
The Scientist's Environment of the Future

A substantial portion of the computing power will be distributed to the scientist's office

Most scene rendering will occur in "real time"

Most simulations will be performed interactively with the scientist observing and steering the simulation in "real time" with highly effective controls

The Scientist's Environment of the Future



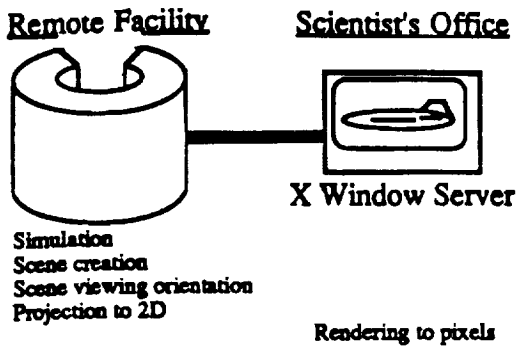
Conclusions

Breakthroughs in workstations have greatly enhanced our capability to experience and understand computer simulations

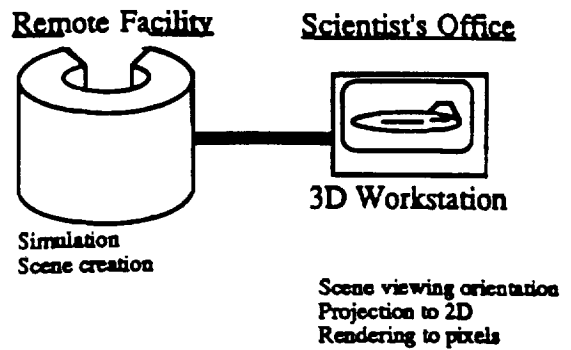
There is still room for major improvements in workstations for scientific analysis.

The most critical task facing scientists doing computer simulations is extraction and display of the key features of the simulations -- more research is required on this task.

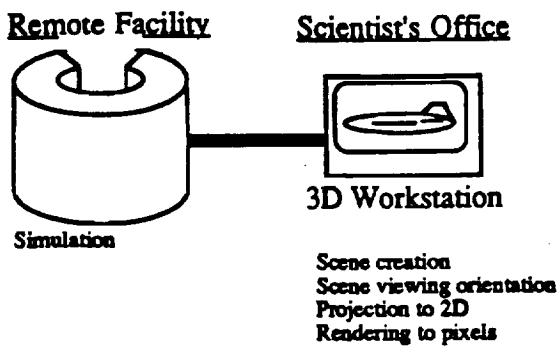
Most of the Work Done on a Supercomputer



Most of the Graphics Done on a Workstation



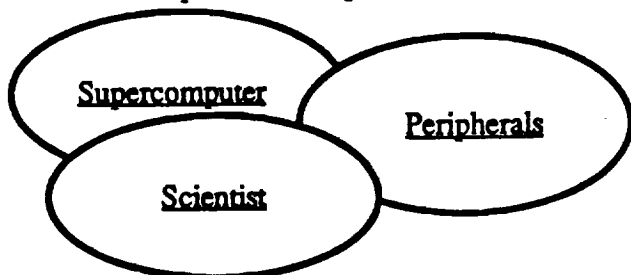
All of the Graphics Done on a Workstation



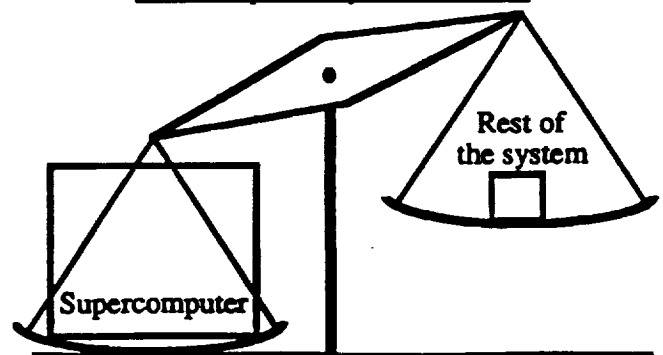
Rationale for Selection of Approach

Selection Criterion

Maximize the performance/price of the total system



Some Supercomputer Centers



Appendix D

**SD/FAST
on the
VAX and AD-100
with an
Example Problem**

Table of Contents

INTRODUCTION.....	D-3
TOOL	D-3
CONSIDERATIONS.....	D-4
MODEL	D-4
PROGRAM.....	D-5
NOTES ON THE SIMULATION	D-7
CONCLUSION	D-8
Input File	D-9
Initialization File	D-12
Dynamics File	D-20

INTRODUCTION

This report is the outcome of a project initiated to evaluate the use of SD/FAST, an equation generating program, in the specialized environment of the AD 100 simulation engine of the Visualization and Simulation Lab of The Research Institute. A program was written to simulate a slewing maneuver of a four body spacecraft model. The model is described on pages 13 to 36 of NASA JPL Technical Report 32-1592, "Attitude Dynamics Simulation Subroutines for Systems of Hinge-Connected Rigid Bodies" by G.E. Fleischer and P.W. Likins. This simulation was converted to ADSIM from the sample program, written in FORTRAN, found in Appendix B of Symbolic Dynamics' SD/FAST User's Manual. Plots were then generated in order to evaluate the performance and accuracy of the ADSIM simulation in this simple problem.

TOOL

SD/FAST is a program used for the generation of dynamic equations from a simple user input file describing the system to be modeled. This program can handle any system that can be described by a series of rigid bodies connected with hinges. The input file consists of descriptions of each body in the system (mass, inertia matrix, reference orientation), joint descriptions, and any constraints added to the system. From this file, the SD/FAST program generates two other files. The first is an initialization file that is used to initialize state variables and set up the environment in which the simulation is to run. The other file generated by SD/FAST is the dynamics file. This file contains all of the equations of motion needed to model the system. These two files generated by SD/FAST do not constitute a complete simulation themselves. The user must add all of the necessary controllers and sensors in his own file. The user must also add any data concerning prescribed motion in any part of the system to be modeled. This information is then used to drive the SD/FAST generated portions of the simulation (particularly the dynamics file).

CONSIDERATIONS

There were many considerations that had to be addressed when converting the sample program from FORTRAN to ADSIM. The first thing that had to be done to realize that the two languages use SD/FAST differently. ADSIM uses the initialization file generated by SD/FAST as a place in which to incorporate its user-written code. The dynamics file generated by SD/FAST is incorporated into this framework. FORTRAN handles the task quite differently. The user includes calls to the initialization and dynamics files. The incorporation of the user's ADSIM code into the SD/FAST generated files makes the ADSIM much smoother and easier to follow. The addition of a sophisticated, variable time-step integrator within ADSIM was another major difference between the two languages. This integrator had to be written into the FORTRAN code of the sample program, but was not needed in ADSIM version. Other considerations that had to be addressed were the differences in the two program languages mechanics. The renaming of variables within the different subroutines of the FORTRAN example had to be traced back through the code in order to decipher which ones needed to be ported to ADSIM and which ones were only needed by the FORTRAN. The global scope of ADSIM variables dismissed the need for renaming these variables within different parts of the ADSIM code. Also, the logic that is used in the controller model of the user-written portion of the FORTRAN simulation had to be incorporated into different areas of ADSIM code since ADSIM deals with procedural statements differently. The mathematics of the controller model also had to be deciphered in order to better understand the dynamics involved with the slew maneuver as commanded by the user-written code. The ADSIM code that was obtained by this conversion still retains most of the flow of the FORTRAN code and should be easy to follow.

MODEL

As seen in the accompanying figure the spacecraft model contains a number of elements (four bodies with ten degrees of freedom). The spacecraft model used during the simulation includes a bus fixed to a boom by means of a u-joint. Atop the bus is a camera attached by means of shaft (or clock). The shaft is connected to the bus by means of a pin joint while the camera is connected to the shaft by another pin joint. The positions of these joints are Q-4 and Q-5, and their rates U-4 and U-5 respectively. These are the joints that have commanded rates during the simulation. The boom is

flexible; and to simulate this, spring and damping torques are applied to the U-joint between it and the bus. The craft is also equipped with three gas jet thrusters and rate sensing gyros to correct bus attitude should it exceed certain limits. The clock and camera can be slewed by means of variable torque motors at each hinge. A controller model in the program drives these motors.

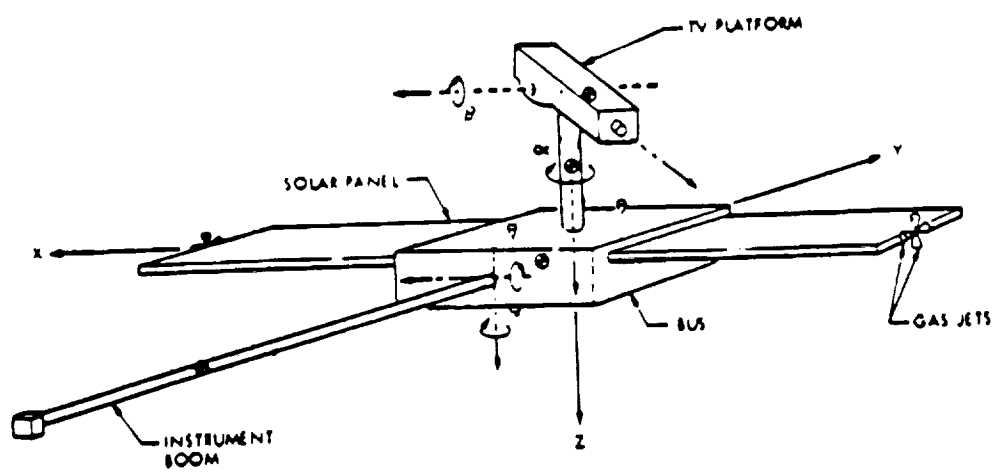
The reference configuration of the clock and the camera is pointing away from the boom with no inclination. At the beginning of the simulation, Q-4, the clock position, is 4 radians from the reference configuration; and Q-5, the camera position, is given to be -0.5 radians. One second into the simulation, the slew will begin and will last for ten seconds. During this time, the clock will cover a 0.25 radian rotation from 4 to 3.75 radians and the camera will cover a 0.10 radian decrease in the declination of the camera from -0.5 to -0.4. The plots of this can be seen in Appendix B.

As the bus drifts, the controller fires the thrusters as necessary in order to correct its attitude. There is a deadband of .0025 radians before this occurs. The effects of these thrusters on the bus and the boom can be seen in the plots in Appendix B.

PROGRAM

Appendix A contains all of the simulation code and all input files associated with the simulation. The first item is the SD/FAST input file that was provided by Symbolic Dynamics. The next two items are the initialization file and dynamics file. Finally, there are several plots that were obtained from the simulation running on the AD 100.

The simulation program is written in ADSIM with much of the initialization occurring in the FORTRAN sections at the beginning of the code. The ADSIM consists of the main user-written body of code and five functions that control each of the three thrusters and the two hinges that follow prescribed motion. The dynamics file is then incorporated into the code with an INCLUDE statement after the main body of code. The blocks are defined below.



The Example Spacecraft

Initialization File:

- FORTTRAN blocks - These blocks set up the variables in FORTRAN. These will be reassigned as ADSIM variables later in the dynamics file. There is also a description of the system from the input file.
- REGION initial - This block initializes the stat variables and all initial values of all other variables used to drive the simulation.
- FUNCTION thrust - These three blocks each control the three thrusters that correct the attitude shifts of the bus.
- FUNCTION camrat - These two function control the commanded rate of the clock and camera as the slew maneuver is simulated.
- DYNAMIC continuous - This block of code contains both the equations for computing the hinge torques and the equations for computing the force applied to the system by each of the three thrusters.

Dynamics File:

- REGION sd-initial - This section converts the FORTRAN variables in the initialization file into ADSIM variables.
- REGION sd-dynamics - This block computes the state derivatives.
- REGION sd-states - This block performs the integration routine.

NOTES ON THE SIMULATION

The simulation environment consisted of a VAX 3100, 3400 and an Applied Dynamics AD-100. The AD-100 features the ADSIM language. The version of ADSIM used is version 7. This version has the ability to import FORTRAN code generated outside of the AD-100 environment. In this instance the VAX 3100 hosted the SD/FAST software. The spacecraft problem was input to the SD/FAST program in the 3100 and it took

27.06 sec of CPU time to create the FORTRAN coded equations of motion. The ADSIM portion of the code was input to the 3100 and then the ADSIM compiled. This compilation took 50.11 seconds of CPU time. The problem was then executed on the AD-100. One run (the one shown in the graphics) was made at 10 times real time. It was then found by experiment that the problem could be run in 96 times real time i.e. it took 0.3125 (30/96) seconds for a total problem time that would take 30 seconds in real time. The default integration scheme called the Adams-Bashford 2 was used in all instances. This is a single pass integration scheme suitable for real time or hardware in the loop simulations. It is one of over a dozen integration schemes selectable in the ADSIM environment (you may, of course, write your own if you wish).

CONCLUSION

After becoming familiar with the mechanics of SD/FAST in regards to ADSIM and converting the sample program from FORTRAN to ADSIM, the worth of a program such as SD/FAST becomes apparent. It gives the programmer more time to concentrate on the design and control of a system without having to worry about producing the long, complicated equations of motion of the system by hand. The time spent learning how to use SD/FAST is more than made up by the savings in man hours and computer time that it will save later. The sample situation, although far from complex, gave a solid performance when running under ADSIM. The relative strengths of the AD 100 and the ADSIM programming language are more than capable of running this program's generated files with almost no loss in performance except in the largest of simulations where runtime performance is of optimum importance. In these cases, SD/FAST is an option, although it may not yield as good results as other methods of deriving equations of motion.

Input File


```
#
# This is an SD/FAST Input File describing the four-body spacecraft
# from Appendix B of the SD/FAST User's Manual. Note that no external
# forces are allowed and that only the base body can have any external
# torques.
```

```
#
language = adsim
body = bus
  mass = 410
  inertia = 115    -14    14
            ?     316   -34.6
            ?     ?     440
  force = 0 0 0
```

```
body = clock inb = bus joint = pin
  mass = 6.8
  inertia = 0.35  0.35  0
  bodyToJoint = 0  0 -0.75
  inbToJoint  = 0  0 -1.5
  pin = 0 0 1
  force = 0 0 0
  torque = 0 0 0
```

```
body = camera inb = clock joint = pin
  mass = 57.5
  inertia = 4.85    0.41  -0.07
            ?      2.2    0.54
            ?      ?      5.5
  bodyToJoint = 0  -0.22  0.2
  inbToJoint  = 0  -0.1  -0.75
  pin = -1 0 0
  force = 0 0 0
  torque = 0 0 0
```

```
body = boom inb = bus joint = ujoint
  mass = 10.7
  inertia = 27.2  0.2  27.2
  bodyToJoint = 0  3.3  0
  inbToJoint  = 0  -1.2  0
  pin = 0 0 1
  pin = 1 0 0
  force = 0 0 0
  torque = 0 0 0
```

```
# @(#)slew.dat 1.1 (Symbolic Dynamics, Inc.) 88/05/02 20:29:46
```

```

#
# This is an SD/FAST Input File describing the four-body spacecraft
# from Appendix B of the SD/FAST User's Manual. Note that no external
# forces are allowed and that only the base body can have any external
# torques.
#
language = adsim
body = bus
  mass = 410
  inertia = 115    -14    14
            ?      316   -34.6
            ?      ?      440
  force = 0 0 0

bodyv = clock  inb = bus  joint = oin
  mass = 6.8
  inertia = 0.35  0.35  0
  bodyToJoint = 0  0  -0.75
  inbToJoint

```

Initialization File

FORTRAN_SECTION declare
FORTRAN

=====

SD/FAST INITIALIZATION

=====

INITIALIZATION ROUTINE

GENERATED 16-AUG-1990 12:38:31.66 BY SD/FAST, ORDER(N) FORMULATION
(SDFAST A.2.14 #20112) ON MACHINE ID 0A000005VS3100
COPYRIGHT (C) 1988 BY SYMBOLIC DYNAMICS, INC.

DESCRIPTION OF SYSTEM:

BODIES	INB	COORDS Q		
#	NAME	BODY	HINGE TYPE	P=PRESCRIBED
1	bus		ATTITUDE	1 2 3
			TRANSLATION	8 9 10
2	clock	1	PIN(1D)	4
3	camera	2	PIN(1D)	5
4	boom	1	U-JOINT(2D)	6 7

THIS ROUTINE SHOULD BE EDITED TO SUPPLY VALUES FOR ALL VARIABLES
WHICH WERE GIVEN AS QUESTION MARKS IN THE INPUT. THESE VARIABLES
ARE SET TO "?" BELOW AND SHOULD BE CHANGED AS APPROPRIATE.

MEANING OF VARIABLES:

IN THE INDICES BELOW, K IS A BODY NUMBER, I AND J SELECT
ELEMENTS OF A VECTOR OR 3X3 MATRIX, AND Q SELECTS A PARTICULAR
PIN OR SLIDER AT A HINGE COUNTING UP FROM INBOARD TO OUTBOARD.

MK(K) -- MASS OF BODY K
I(K,I,J) -- IJ-TH ELEMENT OF INERTIA MATRIX FOR BODY K
L(Q,I) -- I-TH ELEMENT OF PIN VECTOR AT COORDINATE Q. SINCE
Q(4) IS THE 1ST HINGE COORDINATE, L(4,I) SELECTS THE
1ST PIN OR SLIDER.
RK(K,I) -- I-TH ELEMENT OF VECTOR FROM MASS CENTER OF BODY K TO
ITS INBOARD HINGE POINT
RI(K,I) -- I-TH ELEMENT OF VECTOR FROM MASS CENTER OF THE INBOARD
BODY OF BODY K TO THE INBOARD HINGE POINT OF BODY K

DOUBLE PRECISION MK(4),I(4,3,3)
DOUBLE PRECISION L(7,3),RK(4,3)
DOUBLE PRECISION RI(4,3)

COMMON BLOCK FOR SYSTEM PARAMETERS

COMMON /SDCOMN/ MK,I,L,RK,RI

END FORTRAN

FORTRAN_SECTION start
FORTRAN

C
C VALUES FROM INPUT FILE
C

C BODY 1 (bus)
C

MK(1) = 410.0D0
I(1,1,1) = 115.0D0
I(1,1,2) = -14.0D0
I(1,1,3) = 14.0D0
I(1,2,1) = -14.0D0
I(1,2,2) = 316.0D0
I(1,2,3) = -34.6D0
I(1,3,1) = 14.0D0
I(1,3,2) = -34.6D0
I(1,3,3) = 440.0D0

C
C BODY 2 (clock)
C

MK(2) = 6.8D0
I(2,1,1) = 0.35D0
I(2,1,2) = 0.0D0
I(2,1,3) = 0.0D0
I(2,2,1) = 0.0D0
I(2,2,2) = 0.35D0
I(2,2,3) = 0.0D0
I(2,3,1) = 0.0D0
I(2,3,2) = 0.0D0
I(2,3,3) = 0.0D0
RK(2,1) = 0.0D0
RK(2,2) = 0.0D0
RK(2,3) = -0.75D0
RI(2,1) = 0.0D0
RI(2,2) = 0.0D0
RI(2,3) = -1.5D0
L(4,1) = 0.0D0
L(4,2) = 0.0D0
L(4,3) = 1.0D0

C
C BODY 3 (camera)
C

MK(3) = 57.5D0
I(3,1,1) = 4.85D0
I(3,1,2) = 0.41D0
I(3,1,3) = -0.07D0
I(3,2,1) = 0.41D0
I(3,2,2) = 2.2D0
I(3,2,3) = 0.54D0
I(3,3,1) = -0.07D0
I(3,3,2) = 0.54D0
I(3,3,3) = 5.5D0
RK(3,1) = 0.0D0
RK(3,2) = -0.22D0
RK(3,3) = 0.2D0
RI(3,1) = 0.0D0
RI(3,2) = -0.1D0
RI(3,3) = -0.75D0

```

      L(5,1) = -1.0D0
      L(5,2) = 0.0D0
      L(5,3) = 0.0D0
C
C   BODY 4 (boom)
C
      MK(4) = 10.7D0
      I(4,1,1) = 27.2D0
      I(4,1,2) = 0.0D0
      I(4,1,3) = 0.0D0
      I(4,2,1) = 0.0D0
      I(4,2,2) = 0.2D0
      I(4,2,3) = 0.0D0
      I(4,3,1) = 0.0D0
      I(4,3,2) = 0.0D0
      I(4,3,3) = 27.2D0
      RK(4,1) = 0.0D0
      RK(4,2) = 3.3D0
      RK(4,3) = 0.0D0
      RI(4,1) = 0.0D0
      RI(4,2) = -1.2D0
      RI(4,3) = 0.0D0
      L(6,1) = 0.0D0
      L(6,2) = 0.0D0
      L(6,3) = 1.0D0
      L(7,1) = 1.0D0
      L(7,2) = 0.0D0
      L(7,3) = 0.0D0
C
END FORTRAN

FORTRAN_SECTION *

REGION initial

! SET THIS FLAG NON-ZERO WHENEVER YOU WANT TO COMPUTE
! ANGULAR MOMENTUM, KINETIC ENERGY, ETC.

      sdchek flag = 0.0D0

! INITIAL CONDITIONS FOR STATE VARIABLES

      Q_1@ = 0.0d0
      Q_2@ = 0.0d0
      Q_3@ = 0.0d0
      Q_4@ = 4.0d0
      Q_5@ = -0.5d0
      Q_6@ = 0.0d0
      Q_7@ = 0.0d0
      Q_8@ = 0.0d0
      Q_9@ = 0.0d0
      Q_10@ = 0.0d0
      U_1@ = 0.0d0
      U_2@ = 0.0d0
      U_3@ = 0.0d0
      u_4@ = 0.0d0

```

```
U_5@ = 0.0d0
U_6@ = 0.0d0
U_7@ = 0.0d0
U_8@ = 0.0d0
U_9@ = 0.0d0
U_10@ = 0.0d0
```

! EXTERNAL FORCE AND TORQUE INITIALIZATIONS

```
FK_1_1= 0.0D0
FK_1_2= 0.0D0
FK_1_3= 0.0D0
TK_1_1= 0.0d0
TK_1_2= 0.0d0
TK_1_3= 0.0d0
```

```
FK_2_1= 0.0D0
FK_2_2= 0.0D0
FK_2_3= 0.0D0
TK_2_1= 0.0D0
TK_2_2= 0.0D0
TK_2_3= 0.0D0
```

```
FK_3_1= 0.0D0
FK_3_2= 0.0D0
FK_3_3= 0.0D0
TK_3_1= 0.0D0
TK_3_2= 0.0D0
TK_3_3= 0.0D0
```

```
FK_4_1= 0.0D0
FK_4_2= 0.0D0
FK_4_3= 0.0D0
TK_4_1= 0.0D0
TK_4_2= 0.0D0
TK_4_3= 0.0D0
```

! HINGE FORCE OR TORQUE INITIALIZATIONS

```
TAU_4 = 0.0d0
TAU_5 = 0.0d0
TAU_6 = 0.0d0
TAU_7 = 0.0d0
```

! A-FRAME INITIALIZATION

```
WA_1 = 0.0d0
WA_2 = 0.0d0
WA_3 = 0.0d0
VA_1 = 0.0d0
VA_2 = 0.0d0
VA_3 = 0.0d0
```

! USER ADDED VARIABLE ASSIGNMENTS

```
CLKCMD = 4.0D0
```

```

CAMCMD = -0.5D0
TIME = 0.0d0
CLKCMD = 4.0D0
CAMCMD = -0.5D0
qyro = 2.0d0
l_1 = 0.23d0
l_2 = 0.21d0
l_3 = 0.31d0
k_1 = 3500.0d0
k_2 = 3500.0d0
k_3 = 2000.0d0
k_4 = 2000.0d0
b_1 = 20.0d0
b_2 = 20.0d0
b_3 = 10.0d0
b_4 = 10.0d0
END REGION initial

function thrust1(tr1 = time,err_1)

dband = 0.0025d0
tmin = 0.02d0
fire_1 = 0.0d0
toff_1 = 0.0d0

if (err_1.lt.-dband) then
    fire_1 = 1.0d0
    toff_1 = time + tmin
end if

if (err_1.gt.dband) then
    fire_1 = -1.0d0
    toff_1 = time + tmin
end if

if (time.ge.toff_1 ) then
    fire_1 = 0.0d0
end if

tr1 = fire_1

end function

function thrust2(tr2 = time,err_2)

dband = 0.0025d0
tmin = 0.02d0
fire_2 = 0.0d0
toff_2 = 0.0d0

if (err_2.lt.-dband) then
    fire_2 = 1.0d0
    toff_2 = time + tmin
end if

if (err_2.gt.dband) then

```



```

    fire_2      = -1.0d0
    toff_2      = time + tmin
end if

```

```

if (time.ge.toff_2 ) then
    fire_2      = 0.0d0
end if

```

```

tr2 = fire_2

```

```

end function

```

```

function thrust3(tr3 = time,err_3)

```

```

    dband = 0.0025d0
    tmin = 0.02d0
    fire_3 = 0.0d0
    toff_3 = 0.0d0

```

```

if (err_3.lt.-dband) then
    fire_3      = 1.0d0
    toff_3      = time + tmin
end if

```

```

if (err_3.gt.dband) then
    fire_3      = -1.0d0
    toff_3      = time + tmin
end if

```

```

if (time.ge.toff_3 ) then
    fire_3      = 0.0d0
end if

```

```

tr3 = fire_3

```

```

end function

```

```

function cmdrat1(clkcmd = time)

```

```

    STRTSL = 1.0d0
    STOPSL = 11.0d0
    clkkrat = -0.025d0

```

```

if (time.lt.strtsl) then
    clkcmd = 4.0d0
end if

```

```

if ((time.gt.strtsl).and.(time.lt.stopsl)) then
    clkcmd = 4.0d0 + (clkkrat * (time-1.0d0))
end if

```

```

if (time.gt.stopsl) then
    clkcmd = 3.75d0
end if

```

```

end function

```

```

function cmdrat2(camcmd = time)

```

```

      STRTSL = 1.0d0
      STOPSL = 11.0d0
      camrat = 0.01d0
      if (time.lt.strtssl) then
         camcmd = -0.5d0
      end if

      if ((time.gt.strtssl).and.(time.lt.stoosl)) then
         camcmd = -0.5d0 + (camrat * (time-1.0d0))
      end if

      if (time.gt.stoosl) then
         camcmd = -0.4d0
      end if

end function

```

DYNAMIC continuous

```

clkcmd = cmdrat1(time)
camcmd = cmdrat2(time)
time = system_time
tau_4 = -k_1*(q_4 - CLKCMD) - b_1*u_4
tau_5 = -k_2*(q_5 - CAMCMD) - b_2*u_5
tau_6 = -k_3*q_6 - b_3*u_6
tau_7 = -k_4*q_7 - b_4*u_7

```

```

tr1 = thrust1(time,err_1)
err_1 = gyro * u_1 + q_1
tk_1_1 = l_1 * tr1
tr2 = thrust2(time,err_2)
err_2 = gyro * u_2 + q_2
tk_1_2 = l_2 * tr2
tr3 = thrust3(time,err_3)
err_3 = gyro * u_3 + q_3
tk_1_3 = l_3 * tr3

```

ENDDYNAMIC

INCLUDE 'SLEW.DYN'

```

runspecs
  endtime = 30,
  speedup = 10

```

EXECUTE 'SDCONTINUOUS'

Dynamics File

REGION sd_initial

VARIABLES FROM THE SD/FAST INPUT FILE

```

MK_1 = [MK(1)]
I_1_1_1 = [I(1,1,1)]
I_1_1_2 = [I(1,1,2)]
I_1_1_3 = [I(1,1,3)]
I_1_2_1 = [I(1,2,1)]
I_1_2_2 = [I(1,2,2)]
I_1_2_3 = [I(1,2,3)]
I_1_3_1 = [I(1,3,1)]
I_1_3_2 = [I(1,3,2)]
I_1_3_3 = [I(1,3,3)]
MK_2 = [MK(2)]
I_2_1_1 = [I(2,1,1)]
I_2_1_2 = [I(2,1,2)]
I_2_1_3 = [I(2,1,3)]
I_2_2_1 = [I(2,2,1)]
I_2_2_2 = [I(2,2,2)]
I_2_2_3 = [I(2,2,3)]
I_2_3_1 = [I(2,3,1)]
I_2_3_2 = [I(2,3,2)]
I_2_3_3 = [I(2,3,3)]
RK_2_1 = [RK(2,1)]
RK_2_2 = [RK(2,2)]
RK_2_3 = [RK(2,3)]
RI_2_1 = [RI(2,1)]
RI_2_2 = [RI(2,2)]
RI_2_3 = [RI(2,3)]
L_4_1 = [L(4,1)]
L_4_2 = [L(4,2)]
L_4_3 = [L(4,3)]
MK_3 = [MK(3)]
I_3_1_1 = [I(3,1,1)]
I_3_1_2 = [I(3,1,2)]
I_3_1_3 = [I(3,1,3)]
I_3_2_1 = [I(3,2,1)]
I_3_2_2 = [I(3,2,2)]
I_3_2_3 = [I(3,2,3)]
I_3_3_1 = [I(3,3,1)]
I_3_3_2 = [I(3,3,2)]
I_3_3_3 = [I(3,3,3)]
RK_3_1 = [RK(3,1)]
RK_3_2 = [RK(3,2)]
RK_3_3 = [RK(3,3)]
RI_3_1 = [RI(3,1)]
RI_3_2 = [RI(3,2)]
RI_3_3 = [RI(3,3)]
L_5_1 = [L(5,1)]
L_5_2 = [L(5,2)]
L_5_3 = [L(5,3)]
MK_4 = [MK(4)]
I_4_1_1 = [I(4,1,1)]
I_4_1_2 = [I(4,1,2)]
I_4_1_3 = [I(4,1,3)]
```

```

I_4_2_1 = [I(4,2,1)]
I_4_2_2 = [I(4,2,2)]
I_4_2_3 = [I(4,2,3)]
I_4_3_1 = [I(4,3,1)]
I_4_3_2 = [I(4,3,2)]
I_4_3_3 = [I(4,3,3)]
RK_4_1 = [RK(4,1)]
RK_4_2 = [RK(4,2)]
RK_4_3 = [RK(4,3)]
RI_4_1 = [RI(4,1)]
RI_4_2 = [RI(4,2)]
RI_4_3 = [RI(4,3)]
L_6_1 = [L(6,1)]
L_6_2 = [L(6,2)]
L_6_3 = [L(6,3)]
L_7_1 = [L(7,1)]
L_7_2 = [L(7,2)]
L_7_3 = [L(7,3)]

```

END REGION sd_initial

REGION sd_dynamics

```

=====
SUBROUTINE SDNSIM(Q,U,FK,TAU,TK,WA,VA,QDOT,UDOT)
=====

```

```

! THIS IS THE DERIVATIVE SECTION FOR A 4-BODY FREE-FLYING
! SYSTEM WITH 4 HINGE DEGREE(S) OF FREEDOM.
! PLUS 3 ATTITUDE AND 3 TRANSLATIONAL DEGREES OF FREEDOM.

```

```

! GENERATED 16-AUG-1990 12:38:30.91 BY SD/FAST. ORDER(N) FORMULATION
! (SDFAST A.2.14 #20112) ON MACHINE ID 0A000005V53100
! COPYRIGHT (C) 1988 BY SYMBOLIC DYNAMICS, INC.

```

```

! DOUBLE PRECISION Q          (10)
! DOUBLE PRECISION U          (10)
! DOUBLE PRECISION FK         (4,3)
! DOUBLE PRECISION TAU        (7)
! DOUBLE PRECISION TK         (4,3)
! DOUBLE PRECISION WA         (3)
! DOUBLE PRECISION VA         (3)
! DOUBLE PRECISION QDOT       (10)
! DOUBLE PRECISION UDOT       (10)

! DOUBLE PRECISION RK         (4,3)
! DOUBLE PRECISION RI         (4,3)
! DOUBLE PRECISION L          (7,3)
! DOUBLE PRECISION I          (4,3,3)
! DOUBLE PRECISION MK         (4)
! DOUBLE PRECISION IADJ       (5,3,3)
! DOUBLE PRECISION CIK        (5,3,3)
! DOUBLE PRECISION WADOTC     (3)
! DOUBLE PRECISION RSK        (5,3)
! DOUBLE PRECISION MRSK       (5,3)
! DOUBLE PRECISION RIK        (5,3)

```

DOUBLE PRECISION WPK	(7,5,3)
DOUBLE PRECISION VPK	(7,5,3)
DOUBLE PRECISION WK	(5,3)
DOUBLE PRECISION WI	(5,3)
DOUBLE PRECISION TTK	(5,3)
DOUBLE PRECISION FTK	(5,3)
DOUBLE PRECISION OK	(5,3)
DOUBLE PRECISION AK	(5,3)
DOUBLE PRECISION AKI	(5,3)
DOUBLE PRECISION GMK1	(2,2,3,3)
DOUBLE PRECISION OGMK1	(2,2,3,3)
DOUBLE PRECISION GFK1	(2,3)
DOUBLE PRECISION OGFK1	(2,3)
DOUBLE PRECISION ACC1	(3)
DOUBLE PRECISION ALPH1	(3)
DOUBLE PRECISION RIK11	(3,3)
DOUBLE PRECISION WKRI1	(3)
DOUBLE PRECISION WRSK1	(3)
DOUBLE PRECISION GMK2	(2,2,3,3)
DOUBLE PRECISION OGMK2	(2,2,3,3)
DOUBLE PRECISION GFK2	(2,3)
DOUBLE PRECISION OGFK2	(2,3)
DOUBLE PRECISION ACC2	(3)
DOUBLE PRECISION ALPH2	(3)
DOUBLE PRECISION RIKT2	(3,3)
DOUBLE PRECISION WKRI2	(3)
DOUBLE PRECISION WRSK2	(3)
DOUBLE PRECISION GMK3	(2,2,3,3)
DOUBLE PRECISION OGMK3	(2,2,3,3)
DOUBLE PRECISION GFK3	(2,3)
DOUBLE PRECISION OGFK3	(2,3)
DOUBLE PRECISION ACC3	(3)
DOUBLE PRECISION ALPH3	(3)
DOUBLE PRECISION RIKT3	(3,3)
DOUBLE PRECISION WKRI3	(3)
DOUBLE PRECISION WRSK3	(3)
DOUBLE PRECISION GMK4	(2,2,3,3)
DOUBLE PRECISION OGMK4	(2,2,3,3)
DOUBLE PRECISION GFK4	(2,3)
DOUBLE PRECISION OGFK4	(2,3)
DOUBLE PRECISION ACC4	(3)
DOUBLE PRECISION ALPH4	(3)
DOUBLE PRECISION RIKT4	(3,3)
DOUBLE PRECISION WKRI4	(3)
DOUBLE PRECISION WRSK4	(3)
DOUBLE PRECISION GMK5	(2,2,3,3)
DOUBLE PRECISION OGMK5	(2,2,3,3)
DOUBLE PRECISION GFK5	(2,3)
DOUBLE PRECISION OGFK5	(2,3)
DOUBLE PRECISION ACC5	(3)
DOUBLE PRECISION ALPH5	(3)
DOUBLE PRECISION RIKT5	(3,3)
DOUBLE PRECISION WKRI5	(3)
DOUBLE PRECISION WRSK5	(3)
DOUBLE PRECISION LD	(2,2,3,3)
DOUBLE PRECISION DI	(2,2,3,3)

DOUBLE PRECISION	VIM1	(2,3)
DOUBLE PRECISION	TAUP1	
DOUBLE PRECISION	MPP1	
DOUBLE PRECISION	VIP1	(2,3)
DOUBLE PRECISION	VIM2	(2,3)
DOUBLE PRECISION	TAUP2	
DOUBLE PRECISION	MPP2	
DOUBLE PRECISION	VIP2	(2,3)
DOUBLE PRECISION	VIM3	(2,3)
DOUBLE PRECISION	TAUP3	
DOUBLE PRECISION	MPP3	
DOUBLE PRECISION	VIP3	(2,3)
DOUBLE PRECISION	VIM4	(2,3)
DOUBLE PRECISION	TAUP4	
DOUBLE PRECISION	MPP4	
DOUBLE PRECISION	VIP4	(2,3)
DOUBLE PRECISION	VIM5	(2,3)
DOUBLE PRECISION	TAUP5	
DOUBLE PRECISION	MPP5	
DOUBLE PRECISION	VIP5	(2,3)
DOUBLE PRECISION	VIM6	(2,3)
DOUBLE PRECISION	TAUP6	
DOUBLE PRECISION	MPP6	
DOUBLE PRECISION	VIP6	(2,3)
DOUBLE PRECISION	VIM7	(2,3)
DOUBLE PRECISION	TAUP7	
DOUBLE PRECISION	MPP7	
DOUBLE PRECISION	VIP7	(2,3)
DOUBLE PRECISION	SOLN6	(6)
DOUBLE PRECISION	W6	(6)
DOUBLE PRECISION	V6	(6)
DOUBLE PRECISION	L11	(3,3)
DOUBLE PRECISION	L21	(3,3)
DOUBLE PRECISION	L22	(3,3)
DOUBLE PRECISION	D1	(3,3)
DOUBLE PRECISION	D2	(3,3)
DOUBLE PRECISION	D1INV	(3,3)
DOUBLE PRECISION	D2INV	(3,3)
DOUBLE PRECISION	L11D1	(3,3)
DOUBLE PRECISION	D1L21	(3,3)
DOUBLE PRECISION	L22D2	(3,3)
DOUBLE PRECISION	CL11	(3,3)
DOUBLE PRECISION	CL22	(3,3)
DOUBLE PRECISION	RCL	(3,3)
DOUBLE PRECISION	CL11D1	(3,3)
DOUBLE PRECISION	CL22D2	(3,3)
DOUBLE PRECISION	RCLD1	(3,3)
DOUBLE PRECISION	CF	(3)
DOUBLE PRECISION	TEMP(100)	
DOUBLE PRECISION	S1,C1	
DOUBLE PRECISION	S2,C2	
DOUBLE PRECISION	S3,C3	
DOUBLE PRECISION	S4,C4	
DOUBLE PRECISION	S5,C5	
DOUBLE PRECISION	S6,C6	
DOUBLE PRECISION	S7,C7	

! COMPUTE SINES AND COSINES OF Q

```

S1,C1= sin_cos(Q_1)
S2,C2= sin_cos(Q_2)
S3,C3= sin_cos(Q_3)
S4,C4= sin_cos(Q_4)
S5,C5= sin_cos(Q_5)
S6,C6= sin_cos(Q_6)
S7,C7= sin_cos(Q_7)

```

! COMPUTE DIRECTION COSINES

```

CIK_1_1_1= (C2*C3)
CIK_1_1_2= -(S3*C2)
CIK_1_2_1= ((S1*(S2*C3))+(S3*C1))
CIK_1_2_2= ((C1*C3)-(S1*(S2*S3)))
CIK_1_2_3= -(S1*C2)
CIK_1_3_1= ((S1*S3)-(C1*(S2*C3)))
CIK_1_3_2= ((S1*C3)+(C1*(S2*S3)))
CIK_1_3_3= (C1*C2)

```

! COMPUTE QDOT (KINEMATICAL SENS)

```

WADOTC_1= ((CIK_1_3_1*WA_3)+((CIK_1_1_1*WA_1)+(CIK_1_2_1*WA_2)))
WADOTC_2= ((CIK_1_3_2*WA_3)+((CIK_1_1_2*WA_1)+(CIK_1_2_2*WA_2)))
WADOTC_3= ((CIK_1_3_3*WA_3)+((CIK_1_2_3*WA_2)+(WA_1*S2)))
QDOT_1= ((U_1-WADOTC_1)*(C3/C2))+((U_2-WADOTC_2)*(-S3/C2))
QDOT_2= ((S3*(U_1-WADOTC_1))+(C3*(U_2-WADOTC_2)))
QDOT_3= ((U_3-WADOTC_3)+((U_1-WADOTC_1)*(-(S2*C3)/C2))+((U_2-
  WADOTC_2)*((S2*S3)/C2)))
QDOT_4= U_4
QDOT_5= U_5
QDOT_6= U_6
QDOT_7= U_7
QDOT_8= (((U_10*S2)+((CIK_1_1_1*U_8)+(CIK_1_1_2*U_9)))+((Q_9*
  WA_3)-(Q_10*WA_2))-VA_1))
QDOT_9= (((CIK_1_2_3*U_10)+((CIK_1_2_1*U_8)+(CIK_1_2_2*U_9)))+
  ((Q_10*WA_1)-(Q_8*WA_3))-VA_2))
QDOT_10= (((CIK_1_3_3*U_10)+((CIK_1_3_1*U_8)+(CIK_1_3_2*U_9)))+
  ((Q_8*WA_2)-(Q_9*WA_1))-VA_3))

```

! COMPUTE WPK (PARTIAL ANGULAR VELOCITIES)

! COMPUTE WK (ANGULAR VELOCITIES)

! AND WI (U-JOINT PARTIAL ANGULAR VELOCITIES)

```

WK_2_1= ((U_1*C4)+(U_2*S4))
WK_2_2= ((U_2*C4)-(U_1*S4))
WK_2_3= (U_3+U_4)
WK_3_1= (WK_2_1-U_5)
WK_3_2= ((WK_2_2*C5)-(WK_2_3*S5))
WK_3_3= ((WK_2_2*S5)+(WK_2_3*C5))
WK_4_1= ((U_1*C6)+(U_2*S6))
WK_4_2= ((U_2*C6)-(U_1*S6))

```



```

WK_4_3= (U_3+U_6)
WK_5_1= (U_7+WK_4_1)
WK_5_2= ((WK_4_2*C7)+(WK_4_3*S7))
WK_5_3= ((WK_4_3*C7)-(WK_4_2*S7))

```

COMPUTE OK (ANGULAR ACCEL.)

```

OK_2_1= (U_4*WK_2_2)
OK_2_2= -(U_4*WK_2_1)
OK_3_2= ((OK_2_2*C5)-(U_5*WK_3_3))
OK_3_3= ((OK_2_2*S5)+(U_5*WK_3_2))
OK_4_1= (U_6*WK_4_2)
OK_4_2= -(U_6*WK_4_1)
OK_5_2= ((OK_4_2*C7)+(U_7*WK_5_3))
OK_5_3= -((OK_4_2*S7)+(U_7*WK_5_2))

```

COMPUTE VPK (PARTIAL VELOCITIES)

COMPUTE RSK/MRSK

COMPUTE RIK AND IADJ

! COMPUTE AK (LINEAR ACCELERATIONS)

```

AK_1_1= ((U_2*U_10)-(U_3*U_9))
AK_1_2= ((U_3*U_8)-(U_1*U_10))
AK_1_3= ((U_1*U_9)-(U_2*U_8))
AKI_2_1= (AK_1_1-(1.5D0*(U_1*U_3)))
AKI_2_2= (AK_1_2-(1.5D0*(U_2*U_3)))
AKI_2_3= (AK_1_3+(1.5D0*((U_1*U_1)+(U_2*U_2))))
AK_2_1= ((AKI_2_1*C4)+(AKI_2_2*S4))
AK_2_2= ((AKI_2_2*C4)-(AKI_2_1*S4))
AKI_3_1= (AK_2_1-(0.1D0*(WK_2_1*WK_2_2)))
AKI_3_2= (AK_2_2+(0.1D0*((WK_2_1*WK_2_1)+(WK_2_3*WK_2_3)))
AKI_3_3= (AKI_2_3-(0.1D0*(OK_2_1+(WK_2_2*WK_2_3))))
AK_3_2= ((AKI_3_2*C5)-(AKI_3_3*S5))
AK_3_3= ((AKI_3_2*S5)+(AKI_3_3*C5))
AKI_4_1= (AK_1_1-(1.2D0*(U_1*U_2)))
AKI_4_2= (AK_1_2+(1.2D0*((U_1*U_1)+(U_3*U_3)))
AKI_4_3= (AK_1_3-(1.2D0*(U_2*U_3)))
AK_4_1= ((AKI_4_1*C6)+(AKI_4_2*S6))
AK_4_2= ((AKI_4_2*C6)-(AKI_4_1*S6))
AK_5_2= ((AK_4_2*C7)+(AKI_4_3*S7))
AK_5_3= ((AKI_4_3*C7)-(AK_4_2*S7))

```

! COMPUTE FTK AND TTK (INERTIA FORCES AND TORQUES)

```

WRSK3_1= -((0.2D0*WK_3_2)+(0.22D0*WK_3_3))
FTK_2_1= (6.8D0*(AK_2_1+(0.75D0*(OK_2_2+(WK_2_1*WK_2_3))))
FTK_2_2= (6.8D0*(AK_2_2+(0.75D0*((WK_2_2*WK_2_3)-OK_2_1))))
FTK_2_3= (6.8D0*(AKI_2_3-(0.75D0*((WK_2_1*WK_2_1)+(WK_2_2*WK_2_2)
)))
FTK_3_1= (57.5D0*(AKI_3_1+((0.22D0*(WK_3_1*WK_3_2))-0.2D0*(

```

```

WK_3_1*WK_3_3))) - ((0.2D0*OK_3_2) + (0.22D0*OK_3_3))))
FTK_3_2= (57.5D0*(AK_3_2 + ((0.2D0*OK_2_1) + ((WK_3_3*WRSK3_1) - (
0.22D0*(WK_3_1*WK_3_1))))))
FTK_3_3= (57.5D0*(AK_3_3 + ((0.22D0*OK_2_1) + ((0.2D0*(WK_3_1*WK_3_1)
) - (WK_3_2*WRSK3_1))))))
FTK_5_1= (10.7D0*(AK_4_1 + (3.3D0*(OK_5_3 - (WK_5_1*WK_5_2))))))
FTK_5_2= (10.7D0*(AK_5_2 + (3.3D0*((WK_5_1*WK_5_1) + (WK_5_3*WK_5_3)
))))
FTK_5_3= (10.7D0*(AK_5_3 - (3.3D0*(OK_4_1 + (WK_5_2*WK_5_3))))))
TTK_1_1= (((U_2*((440.0D0*U_3) + ((14.0D0*U_1) - (34.6D0*U_2)))) -
(U_3*((316.0D0*U_2) - (14.0D0*U_1) - (34.6D0*U_3)))) - TK_1_1)
TTK_1_2= (((U_3*((14.0D0*U_3) + ((115.0D0*U_1) - (14.0D0*U_2)))) -
(U_1*((440.0D0*U_2) + ((14.0D0*U_1) - (34.6D0*U_2)))) - TK_1_2)
TTK_1_3= (((U_1*((316.0D0*U_2) - (14.0D0*U_1) - (34.6D0*U_3)))) -
(U_2*((14.0D0*U_3) + ((115.0D0*U_1) - (14.0D0*U_2)))) - TK_1_3)
TTK_2_1= ((4.175D0*(OK_2_1 - (WK_2_2*WK_2_3))) - (5.1D0*AK_2_2))
TTK_2_2= ((4.175D0*(OK_2_2 + (WK_2_1*WK_2_3))) + (5.1D0*AK_2_1))
TTK_3_1= (((11.5D0*AK_3_2) + (12.65D0*AK_3_3)) + ((WK_3_2*((8.283D0*
WK_3_3) + ((3.07D0*WK_3_2) - (0.07D0*WK_3_1)))) - (WK_3_3*((3.07D0*
WK_3_3) + ((0.41D0*WK_3_1) + (4.5D0*WK_3_2)))) + ((0.41D0*OK_3_2) +
(9.933D0*OK_2_1) - (0.07D0*OK_3_3)))
TTK_3_2= (((3.07D0*OK_3_3) + ((0.41D0*OK_2_1) + (4.5D0*OK_3_2))) +
((WK_3_3*((0.41D0*WK_3_2) + (9.933D0*WK_3_1)) - (0.07D0*WK_3_3))) -
(WK_3_1*((8.283D0*WK_3_3) + ((3.07D0*WK_3_2) - (0.07D0*WK_3_1)))) -
(11.5D0*AKI_3_1))
TTK_3_3= (((8.283D0*OK_3_3) + ((3.07D0*OK_3_2) - (0.07D0*OK_2_1))) +
((WK_3_1*((3.07D0*WK_3_3) + ((0.41D0*WK_3_1) + (4.5D0*WK_3_2)))) -
(WK_3_2*((0.41D0*WK_3_2) + (9.933D0*WK_3_1)) - (0.07D0*WK_3_3)))) -
(12.65D0*AKI_3_1))
TTK_5_1= (((143.523D0*(WK_5_2*WK_5_3)) + (143.723D0*OK_4_1)) - (
35.31D0*AK_5_3))
TTK_5_3= ((35.31D0*AK_4_1) + ((143.723D0*OK_5_3) - (143.523D0*(
WK_5_1*WK_5_2))))

```

! COMPUTE GMK AND GFK (GENERALIZED MASS AND FORCE)

```

TAUP7= (0.00695782860085D0*(TAU_7 - TTK_5_1))
GFK5_1_3= (FTK_5_3 - (35.31D0*TAUP7))
GFK5_2_1= (TTK_5_1 + (143.723D0*TAUP7))
OGMK4_1_1_2_2= (2.02500643599146D0 + (8.67499356400855D0*(C7*C7))
)
OGMK4_1_1_2_3= (8.67499356400855D0*(S7*C7))
OGMK4_1_1_3_2= OGMK4_1_1_2_3
OGMK4_1_1_3_3= (2.02500643599146D0 + (8.67499356400855D0*(S7*S7))
)
OGMK4_1_2_1_2= -(35.31D0*S7)
OGMK4_1_2_1_3= (35.31D0*C7)
OGMK4_2_1_2_1= -(35.31D0*S7)
OGMK4_2_1_3_1= (35.31D0*C7)
OGMK4_2_2_2_2= (0.2D0 + (143.523D0*(S7*S7)))
OGMK4_2_2_2_3= -(143.523D0*(S7*C7))
OGMK4_2_2_3_2= OGMK4_2_2_2_3
OGMK4_2_2_3_3= (0.2D0 + (143.523D0*(C7*C7)))
CF_2= ((FTK_5_2*C7) - (GFK5_1_3*S7))
CF_3= ((FTK_5_2*S7) + (GFK5_1_3*C7))
OGFK4_1_1= FTK_5_1

```

OGFK4_1_2= CF_2
 OGFK4_1_3= CF_3
 OGFK4_2_1= GFK5_2_1
 OGFK4_2_2= ((0.2D0*(OK_5_2*C7))-(TTK_5_3*S7))
 OGFK4_2_3= ((0.2D0*(OK_5_2*S7))+(TTK_5_3*C7))
 MPP6= (1.0D0/OGMK4_2_2_3_3)
 VIM6_1_1= (MPP6*OGMK4_1_2_1_3)
 VIM6_2_2= (MPP6*OGMK4_2_2_2_3)
 TAUP6= (MPP6*(TAU_6-OGFK4_2_3))
 GFK4_1_1= (OGFK4_1_1+(OGMK4_1_2_1_3*TAUP6))
 GFK4_2_2= (OGFK4_2_2+(OGMK4_2_2_2_3*TAUP6))
 GFK4_2_3= (OGFK4_2_3+(OGMK4_2_2_3_3*TAUP6))
 GMK4_1_1_1_1= (10.7D0-(OGMK4_1_2_1_3*VIM6_1_1))
 GMK4_1_2_1_2= (OGMK4_1_2_1_2-(OGMK4_1_2_1_3*VIM6_2_2))
 GMK4_2_2_2_2= (OGMK4_2_2_2_2-(OGMK4_2_2_2_3*VIM6_2_2))
 TAUP5= (0.100674519279171D0*(TAU_5+TTK_3_1))
 GFK3_1_2= (FTK_3_2-(11.5D0*TAUP5))
 GFK3_1_3= (FTK_3_3-(12.5D0*TAUP5))
 GFK3_2_1= (TTK_3_1-(9.938D0*TAUP5))
 GFK3_2_2= (TTK_3_2-(0.41D0*TAUP5))
 GFK3_2_3= (TTK_3_3+(0.07D0*TAUP5))
 CL11_2_2= (C5-(0.3314555220752D0*S5))
 CL11_3_2= -(S5+(0.3314555220752D0*C5))
 CL22_2_2= (C5+(0.252116283720089D0*S5))
 CL22_3_2= ((0.252116283720089D0*C5)-S5)
 RCL_2_1= -((0.2D0*C5)+(0.22D0*S5))
 RCL_2_2= ((0.001834141228875D0*S5)-(0.010742827197694D0*C5))
 RCL_2_3= ((0.003175257731959D0*S5)-(0.01859793814433D0*C5))
 RCL_3_1= (0.1D0+((0.2D0*S5)-(0.22D0*C5))
 RCL_3_2= ((0.001834141228875D0*C5)+(0.010742827197694D0*S5))
 RCL_3_3= ((0.003175257731959D0*C5)+(0.01859793814433D0*S5))
 OGMK2_1_1_2_2= (6.8D0+((36.5354382287436D0*(S5*S5))+(
 44.1857948253297D0*(CL11_2_2*CL11_2_2))))
 OGMK2_1_1_2_3= ((36.5354382287436D0*(S5*C5))+(
 44.1857948253297D0*(CL11_2_2*CL11_3_2)))
 OGMK2_1_1_3_2= OGMK2_1_1_2_3
 OGMK2_1_1_3_3= (6.8D0+((36.5354382287436D0*(C5*C5))+(
 44.1857948253297D0*(CL11_3_2*CL11_3_2))))
 OGMK2_1_2_1_2= (5.1D0+(57.5D0*RCL_2_1))
 OGMK2_1_2_1_3= (57.5D0*RCL_3_1)
 OGMK2_1_2_2_1= -(5.1D0+((3.65354382287436D0*(S5*C5))+(
 4.41857948253297D0*(CL11_2_2*CL11_3_2))))
 OGMK2_1_2_2_2= ((36.5354382287436D0*(RCL_2_3*S5))+(
 44.1857948253297D0*(CL11_2_2*RCL_2_2)))
 OGMK2_1_2_2_3= ((36.5354382287436D0*(RCL_3_3*S5))+(
 44.1857948253297D0*(CL11_2_2*RCL_3_2)))
 OGMK2_1_2_3_1= -((3.65354382287436D0*(C5*C5))+(
 4.41857948253297D0*(CL11_3_2*CL11_3_2)))
 OGMK2_1_2_3_2= ((36.5354382287436D0*(RCL_2_3*C5))+(
 44.1857948253297D0*(CL11_3_2*RCL_2_2)))
 OGMK2_1_2_3_3= ((36.5354382287436D0*(RCL_3_3*C5))+(
 44.1857948253297D0*(CL11_3_2*RCL_3_2)))
 OGMK2_2_1_1_2= OGMK2_1_2_2_1
 OGMK2_2_1_1_3= OGMK2_1_2_3_1
 OGMK2_2_1_2_1= OGMK2_1_2_1_2
 OGMK2_2_1_2_2= OGMK2_1_2_2_2

OGМК2_2_1_2_3= OGМК2_1_2_3_2
 OGМК2_2_1_3_1= (57.5D0*(RCL_3_1)
 OGМК2_2_1_3_2= OGМК2_1_2_2_3
 OGМК2_2_1_3_3= OGМК2_1_2_3_3
 OGМК2_2_2_1_1= (4.175D0+((0.365354382287436D0*(C5*C5)))+(0.441857948253297D0*(CL11_3_2*CL11_3_2)))
 OGМК2_2_2_1_2= -((3.65354382287436D0*(RCL_2_3*C5)))+(4.41857948253297D0*(CL11_3_2*RCL_2_2))
 OGМК2_2_2_1_3= -((3.65354382287436D0*(RCL_3_3*C5)))+(4.41857948253297D0*(CL11_3_2*RCL_3_2))
 OGМК2_2_2_2_1= OGМК2_2_2_1_2
 OGМК2_2_2_2_2= (4.175D0+((2.16534020618557D0*(CL22_2_2*CL22_2_2)))+(5.36135499290605D0*(S5*S5)))+(36.5354382287436D0*(RCL_2_3*RCL_2_3))+(44.1857948253297D0*(RCL_2_2*RCL_2_2))+(57.5D0*(RCL_2_1*RCL_2_1))
 OGМК2_2_2_2_3= ((2.16534020618557D0*(CL22_2_2*CL22_3_2)))+(5.36135499290605D0*(S5*C5))+(36.5354382287436D0*(RCL_2_3*RCL_3_3))+(44.1857948253297D0*(RCL_2_2*RCL_3_2))+(57.5D0*(RCL_2_1*RCL_3_1))
 OGМК2_2_2_3_1= OGМК2_2_2_1_3
 OGМК2_2_2_3_2= OGМК2_2_2_2_3
 OGМК2_2_2_3_3= ((2.16534020618557D0*(CL22_3_2*CL22_3_2)))+(5.36135499290605D0*(C5*C5))+(36.5354382287436D0*(RCL_3_3*RCL_3_3))+(44.1857948253297D0*(RCL_3_2*RCL_3_2))+(57.5D0*(RCL_3_1*RCL_3_1))
 CF_2= (GFK3_1_2*C5)+(GFK3_1_3*S5)
 CF_3= (GFK3_1_3*C5)-(GFK3_1_2*S5)
 OGFK2_1_1= (FTK_2_1+FTK_3_1)
 OGFK2_1_2= (CF_2+FK_2_2)
 OGFK2_1_3= (CF_3+FTK_2_3)
 OGFK2_2_1= (TTK_2_1+(GFK3_2_1-(0.1D0*CF_3)))
 OGFK2_2_2= (TTK_2_2+(GFK3_2_2*C5)+(GFK3_2_3*S5))
 OGFK2_2_3= ((0.1D0*FTK_3_1)+(GFK3_2_3*C5)-(GFK3_2_2*S5))
 MPP4= (1.0D0/OGМК2_2_2_3_3)
 VIM4_1_1= (MPP4*OGМК2_1_2_1_3)
 VIM4_1_2= (MPP4*OGМК2_1_2_2_3)
 VIM4_1_3= (MPP4*OGМК2_1_2_3_3)
 VIM4_2_1= (MPP4*OGМК2_2_2_1_3)
 VIM4_2_2= (MPP4*OGМК2_2_2_2_3)
 TAUP4= (MPP4*(TAU_4-OGFK2_2_3))
 GFK2_1_1= (OGFK2_1_1+(OGМК2_1_2_1_3*TAUP4))
 GFK2_1_2= (OGFK2_1_2+(OGМК2_1_2_2_3*TAUP4))
 GFK2_1_3= (OGFK2_1_3+(OGМК2_1_2_3_3*TAUP4))
 GFK2_2_1= (OGFK2_2_1+(OGМК2_2_2_1_3*TAUP4))
 GFK2_2_2= (OGFK2_2_2+(OGМК2_2_2_2_3*TAUP4))
 GFK2_2_3= (OGFK2_2_3+(OGМК2_2_2_3_3*TAUP4))
 GМК2_1_1_1_1= (64.3D0-(OGМК2_1_2_1_3*VIM4_1_1))
 GМК2_1_1_1_2= -(OGМК2_1_2_1_3*VIM4_1_2)
 GМК2_1_1_1_3= -(OGМК2_1_2_1_3*VIM4_1_3)
 GМК2_1_1_2_2= (OGМК2_1_1_2_2-(OGМК2_1_2_2_3*VIM4_1_2))
 GМК2_1_1_2_3= (OGМК2_1_1_2_3-(OGМК2_1_2_2_3*VIM4_1_3))
 GМК2_1_1_3_3= (OGМК2_1_1_3_3-(OGМК2_1_2_3_3*VIM4_1_3))
 GМК2_1_2_1_1= -(OGМК2_1_2_1_3*VIM4_2_1)
 GМК2_1_2_1_2= (OGМК2_1_2_1_2-(OGМК2_1_2_1_3*VIM4_2_2))
 GМК2_1_2_2_1= (OGМК2_1_2_2_1-(OGМК2_1_2_2_3*VIM4_2_1))
 GМК2_1_2_2_2= (OGМК2_1_2_2_2-(OGМК2_1_2_2_3*VIM4_2_2))

```

GMK2_1_2_3_1= (OGMK2_1_2_3_1-(OGMK2_1_2_3_3*VIM4_2_1))
GMK2_1_2_3_2= (OGMK2_1_2_3_2-(OGMK2_1_2_3_3*VIM4_2_2))
GMK2_2_2_1_1= (OGMK2_2_2_1_1-(OGMK2_2_2_1_3*VIM4_2_1))
GMK2_2_2_1_2= (OGMK2_2_2_1_2-(OGMK2_2_2_1_3*VIM4_2_2))
GMK2_2_2_2_2= (OGMK2_2_2_2_2-(OGMK2_2_2_2_3*VIM4_2_2))
D1INV_1_1= 0.0D0
if (GMK2_1_1_1_1 .ne. 0.0D0) then
    D1INV_1_1= (1.0D0/GMK2_1_1_1_1)
endif
L11_2_1= (D1INV_1_1*GMK2_1_1_1_2)
L11_3_1= (D1INV_1_1*GMK2_1_1_1_3)
L11D1_2_2= (GMK2_1_1_2_2-(GMK2_1_1_1_2*L11_2_1))
L11D1_3_2= (GMK2_1_1_2_3-(GMK2_1_1_1_3*L11_2_1))
L11D1_3_3= (GMK2_1_1_3_3-(GMK2_1_1_1_3*L11_3_1))
D1INV_2_2= 0.0D0
if (L11D1_2_2 .ne. 0.0D0) then
    D1INV_2_2= (1.0D0/L11D1_2_2)
endif
L11_3_2= (D1INV_2_2*L11D1_3_2)
L11D1_3_3= (L11D1_3_3-(L11_3_2*L11D1_3_2))
D1INV_3_3= 0.0D0
if (L11D1_3_3 .ne. 0.0D0) then
    D1INV_3_3= (1.0D0/L11D1_3_3)
endif
D1L21_2_1= (GMK2_1_2_2_1-(GMK2_1_2_1_1*L11_2_1))
D1L21_2_2= (GMK2_1_2_2_2-(GMK2_1_2_1_2*L11_2_1))
D1L21_3_1= (GMK2_1_2_3_1-(D1L21_2_1*L11_3_2)+(GMK2_1_2_1_1*
    L11_3_1))
D1L21_3_2= (GMK2_1_2_3_2-(D1L21_2_2*L11_3_2)+(GMK2_1_2_1_2*
    L11_3_1))
L21_1_1= (D1INV_1_1*GMK2_1_2_1_1)
L21_1_2= (D1INV_2_2*D1L21_2_1)
L21_1_3= (D1INV_3_3*D1L21_3_1)
L21_2_1= (D1INV_1_1*GMK2_1_2_1_2)
L21_2_2= (D1INV_2_2*D1L21_2_2)
L21_2_3= (D1INV_3_3*D1L21_3_2)
L22D2_1_1= (GMK2_2_2_1_1-(D1L21_3_1*L21_1_3)+(D1L21_2_1*
    L21_1_2)+(GMK2_1_2_1_1*L21_1_1))
L22D2_2_1= (GMK2_2_2_1_2-(D1L21_3_1*L21_2_3)+(D1L21_2_1*
    L21_2_2)+(GMK2_1_2_1_1*L21_2_1))
L22D2_2_2= (GMK2_2_2_2_2-(D1L21_3_2*L21_2_3)+(D1L21_2_2*
    L21_2_2)+(GMK2_1_2_1_2*L21_2_1))
D2INV_1_1= 0.0D0
if (L22D2_1_1 .ne. 0.0D0) then
    D2INV_1_1= (1.0D0/L22D2_1_1)
endif
L22_2_1= (D2INV_1_1*L22D2_2_1)
L22D2_2_2= (L22D2_2_2-(L22_2_1*L22D2_2_1))
D2INV_2_2= 0.0D0
if (L22D2_2_2 .ne. 0.0D0) then
    D2INV_2_2= (1.0D0/L22D2_2_2)
endif
LO_1_1_2_1= L11_2_1
LO_1_1_3_1= L11_3_1
LO_1_1_3_2= L11_3_2
LO_2_1_1_1= L21_1_1

```

ORIGINAL PAGE IS
OF POOR QUALITY

LO_2_1_1_2= L21_1_2
 LO_2_1_1_3= L21_1_3
 LO_2_1_2_1= L21_2_1
 LO_2_1_2_2= L21_2_2
 LO_2_1_2_3= L21_2_3
 LO_2_2_2_1= L22_2_1
 DI_1_1_1_1= GMK2_1_1_1_1
 DI_1_1_2_2= L11D1_2_2
 DI_1_1_3_3= L11D1_3_3
 DI_2_2_1_1= L22D2_1_1
 DI_2_2_2_2= L22D2_2_2
 CL11_1_1= (C4-(LO_1_1_2_1*S4))
 CL11_2_1= (S4+(LO_1_1_2_1*C4))
 CL22_1_1= (C4-(LO_2_2_2_1*S4))
 CL22_2_1= (S4+(LO_2_2_2_1*C4))
 RCL_1_1= ((1.5D0*CL11_2_1)+(LO_2_1_1_1*C4)-(LO_2_1_2_1*S4))
 RCL_1_2= ((1.5D0*C4)+(LO_2_1_1_2*C4)-(LO_2_1_2_2*S4))
 RCL_1_3= ((LO_2_1_1_3*C4)-(LO_2_1_2_3*S4))
 RCL_2_1= ((LO_2_1_1_1*S4)+(LO_2_1_2_1*C4)-(1.5D0*CL11_1_1))
 RCL_2_2= ((1.5D0*S4)+(LO_2_1_1_2*S4)+(LO_2_1_2_2*C4))
 RCL_2_3= ((LO_2_1_1_3*S4)+(LO_2_1_2_3*C4))
 CL11D1_1_1= (CL11_1_1*DI_1_1_1_1)
 CL11D1_1_2= -(DI_1_1_2_2*S4)
 CL11D1_2_1= (CL11_2_1*DI_1_1_1_1)
 CL11D1_2_2= (DI_1_1_2_2*C4)
 CL11D1_3_1= (DI_1_1_1_1*LO_1_1_3_1)
 CL11D1_3_2= (DI_1_1_2_2*LO_1_1_3_2)
 CL22D2_1_1= (CL22_1_1*DI_2_2_1_1)
 CL22D2_1_2= -(DI_2_2_2_2*S4)
 CL22D2_2_1= (CL22_2_1*DI_2_2_1_1)
 CL22D2_2_2= (DI_2_2_2_2*C4)
 RCLD1_1_1= (DI_1_1_1_1*RCL_1_1)
 RCLD1_1_2= (DI_1_1_2_2*RCL_1_2)
 RCLD1_1_3= (DI_1_1_3_3*RCL_1_3)
 RCLD1_2_1= (DI_1_1_1_1*RCL_2_1)
 RCLD1_2_2= (DI_1_1_2_2*RCL_2_2)
 RCLD1_2_3= (DI_1_1_3_3*RCL_2_3)
 OGMK1_1_1_1_1= (410.0D0+((CL11_1_1*CL11D1_1_1)-(CL11D1_1_2*S4))
 OGMK1_1_1_1_2= ((CL11_2_1*CL11D1_1_1)+(CL11D1_1_2*C4))
 OGMK1_1_1_1_3= ((CL11D1_1_1*LO_1_1_3_1)+(CL11D1_1_2*LO_1_1_3_2))
 OGMK1_1_1_2_1= OGMK1_1_1_1_2
 OGMK1_1_1_2_2= (410.0D0+((CL11_2_1*CL11D1_2_1)+(CL11D1_2_2*C4))
 OGMK1_1_1_2_3= ((CL11D1_2_1*LO_1_1_3_1)+(CL11D1_2_2*LO_1_1_3_2))
 OGMK1_1_1_3_1= OGMK1_1_1_1_3
 OGMK1_1_1_3_2= OGMK1_1_1_2_3
 OGMK1_1_1_3_3= (410.0D0+(DI_1_1_3_3+((CL11D1_3_1*LO_1_1_3_1)+
 (CL11D1_3_2*LO_1_1_3_2)))
 OGMK1_1_2_1_1= ((CL11_1_1*RCLD1_1_1)-(RCLD1_1_2*S4))
 OGMK1_1_2_1_2= ((CL11_1_1*RCLD1_2_1)-(RCLD1_2_2*S4))
 OGMK1_1_2_2_1= ((CL11_2_1*RCLD1_1_1)+(RCLD1_1_2*C4))
 OGMK1_1_2_2_2= ((CL11_2_1*RCLD1_2_1)+(RCLD1_2_2*C4))
 OGMK1_1_2_3_1= (RCLD1_1_3+((LO_1_1_3_1*RCLD1_1_1)+(LO_1_1_3_2*
 RCLD1_1_2)))
 OGMK1_1_2_3_2= (RCLD1_2_3+((LO_1_1_3_1*RCLD1_2_1)+(LO_1_1_3_2*
 RCLD1_2_2)))
 OGMK1_2_1_1_1= OGMK1_1_2_1_1

```

OGMK1_2_1_1_2= OGMK1_1_2_2_1
OGMK1_2_1_1_3= OGMK1_1_2_3_1
OGMK1_2_1_2_1= OGMK1_1_2_1_2
OGMK1_2_1_2_2= OGMK1_1_2_2_2
OGMK1_2_1_2_3= OGMK1_1_2_3_2
OGMK1_2_2_1_1= ((115.0D0+(((CL22D2_1_1*CL22D2_1_1)-(CL22D2_1_2*S4))
+((RCL_1_3*RCLD1_1_3)+((RCL_1_1*RCLD1_1_1)+(RCL_1_2*RCLD1_1_2))
)))
OGMK1_2_2_1_2= (((((CL22D2_1_1*CL22D2_1_1)+(CL22D2_1_2*C4))+((
RCL_2_3*RCLD1_1_3)+((RCL_2_1*RCLD1_1_1)+(RCL_2_2*RCLD1_1_2))))-
14.0D0)
OGMK1_2_2_2_1= OGMK1_2_2_1_2
OGMK1_2_2_2_2= (316.0D0+(((CL22D2_1_1*CL22D2_2_1)+(CL22D2_2_2*C4))
+((RCL_2_3*RCLD1_2_3)+((RCL_2_1*RCLD1_2_1)+(RCL_2_2*RCLD1_2_2))
)))
CF_1= ((GFK2_1_1*C4)-(GFK2_1_2*S4))
CF_2= ((GFK2_1_1*S4)+(GFK2_1_2*C4))
OGFK1_1_1= (CF_1+(410.0D0*AK_1_1))
UGFK1_1_2= (CF_2+(410.0D0*AK_1_2))
OGFK1_1_3= (GFK2_1_3+(410.0D0*AK_1_3))
UGFK1_2_1= (TTK_1_1+((1.5D0*CF_2)+(GFK2_2_1*C4)-(GFK2_2_2*S4))
))
UGFK1_2_2= (TTK_1_2+((GFK2_2_1*S4)+(GFK2_2_2*C4))-(1.5D0*CF_1)
))
OGFK1_2_3= (GFK2_2_3+TTK_1_3)
DIINV_1_1= 0.0D0
if (GMK4_1_1_1_1 .ne. 0.0D0) then
    DIINV_1_1= (1.0D0/GMK4_1_1_1_1)
endif
DIINV_2_2= 0.0D0
if (OGMK4_1_1_2_2 .ne. 0.0D0) then
    DIINV_2_2= (1.0D0/OGMK4_1_1_2_2)
endif
L11_3_2= (DIINV_2_2*OGMK4_1_1_2_3)
L11D1_3_3= (OGMK4_1_1_3_3-(L11_3_2*OGMK4_1_1_2_3))
DIINV_3_3= 0.0D0
if (L11D1_3_3 .ne. 0.0D0) then
    DIINV_3_3= (1.0D0/L11D1_3_3)
endif
L21_2_1= (DIINV_1_1*GMK4_1_2_1_2)
L22D2_2_2= (GMK4_2_2_2_2-(GMK4_1_2_1_2*L21_2_1))
D2INV_2_2= 0.0D0
if (L22D2_2_2 .ne. 0.0D0) then
    D2INV_2_2= (1.0D0/L22D2_2_2)
endif
LO_1_1_3_2= L11_3_2
LO_2_1_2_1= L21_2_1
DI_1_1_1_1= GMK4_1_1_1_1
DI_1_1_2_2= OGMK4_1_1_2_2
DI_1_1_3_3= L11D1_3_3
DI_2_2_2_2= L22D2_2_2
RCL_1_1= -(LO_2_1_2_1*S6)
RCL_2_1= (LO_2_1_2_1*C6)
CL11D1_1_1= (DI_1_1_1_1*C6)
CL11D1_1_2= -(DI_1_1_2_2*S6)
CL11D1_2_1= (DI_1_1_1_1*S6)

```

$CL11D1_2_2 = (DI_1_1_1_1 + C6)$
 $CL11D1_3_2 = (DI_1_1_2_2 + LO_1_1_3_2)$
 $CL22D2_1_2 = -(DI_2_2_2_2 * S6)$
 $CL21D2_2_2 = (DI_2_2_2_2 + C6)$
 $RCLD1_1_1 = (DI_1_1_1_1 + RCL_1_1)$
 $RCLD1_1_2 = -(1.2D0 * (DI_1_1_2_2 + LO_1_1_3_2))$
 $RCLD1_2_1 = (DI_1_1_1_1 + RCL_2_1)$
 $RCLD1_3_1 = (1.2D0 * (DI_1_1_1_1 + C6))$
 $RCLD1_3_2 = -(1.2D0 * (DI_1_1_2_2 * S6))$
 $OGMK1_1_1_1_1 = (OGMK1_1_1_1_1 + ((CL11D1_1_1 * C6) - (CL11D1_1_2 * S6)))$
 $OGMK1_1_1_1_2 = (OGMK1_1_1_1_2 + ((CL11D1_1_1 * S6) + (CL11D1_1_2 * C6)))$
 $OGMK1_1_1_1_3 = (OGMK1_1_1_1_3 + (CL11D1_1_2 * LO_1_1_3_2))$
 $OGMK1_1_1_2_1 = OGMK1_1_1_1_2$
 $OGMK1_1_1_2_2 = (OGMK1_1_1_2_2 + ((CL11D1_2_1 * S6) + (CL11D1_2_2 * C6)))$
 $OGMK1_1_1_2_3 = (OGMK1_1_1_2_3 + (CL11D1_2_2 * LO_1_1_3_2))$
 $OGMK1_1_1_3_1 = OGMK1_1_1_1_3$
 $OGMK1_1_1_3_2 = OGMK1_1_1_2_3$
 $OGMK1_1_1_3_3 = (OGMK1_1_1_3_3 + (DI_1_1_3_3 + (CL11D1_3_2 * LO_1_1_3_2)))$
 $OGMK1_1_2_1_1 = (OGMK1_1_2_1_1 + ((RCLD1_1_1 * C6) - (RCLD1_1_2 * S6)))$
 $OGMK1_1_2_1_2 = (OGMK1_1_2_1_2 + (RCLD1_2_1 * C6))$
 $OGMK1_1_2_1_3 = ((RCLD1_3_1 * C6) - (RCLD1_3_2 * S6))$
 $OGMK1_1_2_2_1 = (OGMK1_1_2_2_1 + ((RCLD1_1_1 * S6) + (RCLD1_1_2 * C6)))$
 $OGMK1_1_2_2_2 = (OGMK1_1_2_2_2 + (RCLD1_2_1 * S6))$
 $OGMK1_1_2_2_3 = ((RCLD1_3_1 * S6) + (RCLD1_3_2 * C6))$
 $OGMK1_1_2_3_1 = (OGMK1_1_2_3_1 + (LO_1_1_3_2 * RCLD1_1_2) - (1.2D0 * DI_1_1_3_3))$
 $OGMK1_1_2_3_2 = OGMK1_1_2_3_2$
 $OGMK1_1_2_3_3 = (LO_1_1_3_2 * RCLD1_3_2)$
 $OGMK1_2_1_1_1 = OGMK1_1_2_1_1$
 $OGMK1_2_1_1_2 = OGMK1_1_2_2_1$
 $OGMK1_2_1_1_3 = OGMK1_1_2_3_1$
 $OGMK1_2_1_2_1 = OGMK1_1_2_1_2$
 $OGMK1_2_1_2_2 = OGMK1_1_2_2_2$
 $OGMK1_2_1_2_3 = OGMK1_1_2_3_2$
 $OGMK1_2_1_3_1 = OGMK1_1_2_1_3$
 $OGMK1_2_1_3_2 = OGMK1_1_2_2_3$
 $OGMK1_2_1_3_3 = OGMK1_1_2_3_3$
 $OGMK1_2_2_1_1 = (OGMK1_2_2_1_1 + (((1.44D0 * DI_1_1_3_3) + ((RCL_1_1 * RCLD1_1_1) - (1.2D0 * (LO_1_1_3_2 * RCLD1_1_2)))) - (CL22D2_1_2 * S6)))$
 $OGMK1_2_2_1_2 = (OGMK1_2_2_1_2 + ((CL22D2_1_2 * C6) + (RCL_2_1 * RCLD1_1_1)))$
 $OGMK1_2_2_1_3 = (14.0D0 + (1.2D0 * ((RCLD1_1_1 * C6) - (RCLD1_1_2 * S6))))$
 $OGMK1_2_2_2_1 = OGMK1_2_2_1_2$
 $OGMK1_2_2_2_2 = (OGMK1_2_2_2_2 + ((CL22D2_2_2 * C6) + (RCL_2_1 * RCLD1_2_1)))$
 $OGMK1_2_2_2_3 = ((1.2D0 * (RCLD1_3_1 * C6)) - 34.6D0)$
 $OGMK1_2_2_3_1 = OGMK1_2_2_1_3$
 $OGMK1_2_2_3_2 = OGMK1_2_2_2_3$
 $OGMK1_2_2_3_3 = (440.0D0 + (1.2D0 * ((RCLD1_3_1 * C6) - (RCLD1_3_2 * S6))))$
 $CF_1 = ((GFK4_1_1 * C6) - (OGFK4_1_2 * S6))$
 $CF_2 = ((GFK4_1_1 * S6) + (OGFK4_1_2 * C6))$


```

OGFK1_1_1= (CF_1+OGFK1_1_1)
OGFK1_1_2= (CF_2+OGFK1_1_2)
OGFK1_1_3= (OGFK1_1_3+OGFK4_1_3)
OGFK1_2_1= (OGFK1_2_1+(((OGFK4_2_1*CB)-(OGFK4_2_2*SB))-(1.2D0*
OGFK4_1_3)))
OGFK1_2_2= (OGFK1_2_2+(OGFK4_2_2*CB)+(OGFK4_2_1*SB))
OGFK1_2_3= (OGFK1_2_3+OGFK4_2_3+(1.2D0*CF_1))

```

SOLVE FOR UDOT (ACCELERATIONS)

```

D1INV_1_1= (1.0D0/OGMK1_1_1_1_1)
L11_2_1= (D1INV_1_1*OGMK1_1_1_2_1)
L11_3_1= (D1INV_1_1*OGMK1_1_1_3_1)
L11D1_2_2= (OGMK1_1_1_2_2-(L11_2_1*OGMK1_1_1_2_1))
L11D1_3_2= (OGMK1_1_1_3_2-(L11_2_1*OGMK1_1_1_3_1))
L11D1_3_3= (OGMK1_1_1_3_3-(L11_3_1*OGMK1_1_1_3_1))
D1INV_2_2= (1.0D0/L11D1_2_2)
L11_3_2= (D1INV_2_2*L11D1_3_2)
L11D1_3_3= (L11D1_3_3-(L11_3_2*L11D1_3_2))
D1INV_3_3= (1.0D0/L11D1_3_3)
D1L21_2_1= (OGMK1_1_2_2_1-(L11_2_1*OGMK1_1_2_1_1))
D1L21_2_2= (OGMK1_1_2_2_2-(L11_2_1*OGMK1_1_2_1_2))
D1L21_2_3= (OGMK1_1_2_2_3-(L11_2_1*OGMK1_1_2_1_3))
D1L21_3_1= (OGMK1_1_2_3_1-((D1L21_2_1*L11_3_2)+(L11_3_1*
OGMK1_1_2_1_1)))
D1L21_3_2= (OGMK1_1_2_3_2-((D1L21_2_2*L11_3_2)+(L11_3_1*
OGMK1_1_2_1_2)))
D1L21_3_3= (OGMK1_1_2_3_3-((D1L21_2_3*L11_3_2)+(L11_3_1*
OGMK1_1_2_1_3)))
L21_1_1= (D1INV_1_1*OGMK1_1_2_1_1)
L21_1_2= (D1INV_1_2*D1L21_2_1)
L21_1_3= (D1INV_3_3*D1L21_3_1)
L21_2_1= (D1INV_1_1*OGMK1_1_2_1_2)
L21_2_2= (D1INV_2_2*D1L21_2_2)
L21_2_3= (D1INV_3_3*D1L21_3_2)
L21_3_1= (D1INV_1_1*OGMK1_1_2_1_3)
L21_3_2= (D1INV_2_2*D1L21_2_3)
L21_3_3= (D1INV_3_3*D1L21_3_3)
L22D2_1_1= (OGMK1_2_2_1_1-((D1L21_3_1*L21_1_3)+(D1L21_2_1*
L21_1_2)+(L21_1_1*OGMK1_1_2_1_1)))
L22D2_2_1= (OGMK1_2_2_2_1-((D1L21_3_1*L21_2_3)+(D1L21_2_1*
L21_2_2)+(L21_2_1*OGMK1_1_2_1_1)))
L22D2_2_2= (OGMK1_2_2_2_2-((D1L21_3_2*L21_2_3)+(D1L21_2_2*
L21_2_2)+(L21_2_1*OGMK1_1_2_1_2)))
L22D2_3_1= (OGMK1_2_2_3_1-((D1L21_3_1*L21_3_3)+(D1L21_2_1*
L21_3_2)+(L21_3_1*OGMK1_1_2_1_1)))
L22D2_3_2= (OGMK1_2_2_3_2-((D1L21_3_2*L21_3_3)+(D1L21_2_2*
L21_3_2)+(L21_3_1*OGMK1_1_2_1_2)))
L22D2_3_3= (OGMK1_2_2_3_3-((D1L21_3_3*L21_3_3)+(D1L21_2_3*
L21_3_2)+(L21_3_1*OGMK1_1_2_1_3)))
D2INV_1_1= (1.0D0/L22D2_1_1)
L22_2_1= (D2INV_1_1*L22D2_2_1)
L22_3_1= (D2INV_1_1*L22D2_3_1)
L22D2_2_2= (L22D2_2_2-(L22_2_1*L22D2_2_1))
L22D2_3_2= (L22D2_3_2-(L22_2_1*L22D2_3_1))
L22D2_3_3= (L22D2_3_3-(L22_3_1*L22D2_3_1))

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

D2INV_2_2= (1.0D0/L22D2_2_2)
L22_3_2= (D2INV_2_2*L22D2_3_2)
L22D2_3_3= (L22D2_3_3-(L22_3_2*L22D2_3_2))
LO_1_1_2_1= L11_2_1
LO_1_1_3_1= L11_3_1
LO_1_1_3_2= L11_3_2
LO_2_1_1_1= L21_1_1
LO_2_1_1_2= L21_1_2
LO_2_1_1_3= L21_1_3
LO_2_1_2_1= L21_2_1
LO_2_1_2_2= L21_2_2
LO_2_1_2_3= L21_2_3
LO_2_1_3_1= L21_3_1
LO_2_1_3_2= L21_3_2
LO_2_1_3_3= L21_3_3
LO_2_2_2_1= L22_2_1
LO_2_2_3_1= L22_3_1
LO_2_2_3_2= L22_3_2
DI_1_1_1_1= OGMK1_1_1_1_1
DI_1_1_2_2= L11D1_2_2
DI_1_1_3_3= L11D1_3_3
DI_2_2_1_1= L22D2_1_1
DI_2_2_2_2= L22D2_2_2
DI_2_2_3_3= L22D2_3_3
W6_2= (((LO_1_1_2_1*OGFK1_1_1)-OGFK1_1_2)
W6_3= (((LO_1_1_3_1*OGFK1_1_1)-OGFK1_1_3)-(LO_1_1_3_2*W6_2))
W6_4= (((LO_2_1_1_1*OGFK1_1_1)-OGFK1_2_1)-(LO_2_1_1_2*W6_2))-
      (LO_2_1_1_3*W6_3))
W6_5= (((LO_2_1_2_1*OGFK1_1_1)-OGFK1_2_2)-(LO_2_1_2_2*W6_2))-
      (LO_2_1_2_3*W6_3)-(LO_2_2_2_1*W6_4))
W6_6= (((LO_2_1_3_1*OGFK1_1_1)-OGFK1_2_3)-(LO_2_1_3_2*W6_2))-
      (LO_2_1_3_3*W6_3)-(LO_2_2_3_1*W6_4)-(LO_2_2_3_2*W6_5))
V6_1= (-OGFK1_1_1/DI_1_1_1_1)
V6_2= (W6_2/DI_1_1_2_2)
V6_3= (W6_3/DI_1_1_3_3)
V6_4= (W6_4/DI_2_2_1_1)
V6_5= (W6_5/DI_2_2_2_2)
V6_6= (W6_6/DI_2_2_3_3)
SOLN6_6= V6_6
SOLN6_5= (V6_5-(LO_2_2_3_2*SOLN6_6))
SOLN6_4= ((V6_4-(LO_2_2_2_1*SOLN6_5)-(LO_2_2_3_1*SOLN6_6))
SOLN6_3= (((V6_3-(LO_2_1_1_3*SOLN6_4)-(LO_2_1_2_3*SOLN6_5))-
      (LO_2_1_3_3*SOLN6_6))
SOLN6_2= (((V6_2-(LO_1_1_3_2*SOLN6_3)-(LO_2_1_1_2*SOLN6_4))-
      (LO_2_1_2_2*SOLN6_5)-(LO_2_1_3_2*SOLN6_6))
SOLN6_1= (((V6_1-(LO_1_1_2_1*SOLN6_2)-(LO_1_1_3_1*SOLN6_3))-
      (LO_2_1_1_1*SOLN6_4)-(LO_2_1_2_1*SOLN6_5)-(LO_2_1_3_1*
      SOLN6_6))
UDOT_1= SOLN6_4
UDOT_2= SOLN6_5
UDOT_3= SOLN6_6
UDOT_8= SOLN6_1
UDOT_9= SOLN6_2
UDOT_10= SOLN6_3
ALPH2_1= ((UDOT_1*C4)+(UDOT_2*S4))
ALPH2_2= ((UDOT_2*C4)-(UDOT_1*S4))

```

```

ACC2_1= ((S4*(UDOT_9+(1.5D0*UDOT_10)))+(C4*(UDOT_8-(1.5D0*
UDOT_2))))
ACC2_2= ((C4*(UDOT_9+(1.5D0*UDOT_10)))-(S4*(UDOT_8-(1.5D0*
UDOT_2))))
UDOT_4=((taup4-((udot_3+((alph2_1*vim4_2_1)+(alph2_2*vim4_2_2)))+(
((udot_10*vim4_1_3)+((acc2_1*vim4_1_1)+(acc2_2*vim4_1_2))))))
ALPH3_2= ((ALPH2_2*C5)-(S5*(UDOT_3+UDOT_4)))
ALPH3_3= ((ALPH2_2*S5)+(C5*(UDOT_3+UDOT_4)))
ACC3_1= (ACC2_1+(0.1D0*(UDOT_3+UDOT_4)))
ACC3_2= ((ACC2_2*C5)-(S5*(UDOT_10-(0.1D0*ALPH2_1))))
ACC3_3= ((ACC2_2*S5)+(C5*(UDOT_10-(0.1D0*ALPH2_1))))
UDOT_5=((taup5-(((0.007047216349542d0*alpa3_3)-(alph2_1+(
0.04127655290446d0*alph3_2)))-((1.15775697171046d0*acc3_2)+(
(1.27353266888151d0*acc3_3))))))
ALPH4_1= ((UDOT_1*C6)+(UDOT_2*S6))
ALPH4_2= ((UDOT_2*C6)-(UDOT_1*S6))
ACC4_1= ((UDOT_9*S6)+(C6*(UDOT_8+(1.2D0*UDOT_3))))
ACC4_2= ((UDOT_9*C6)-(S6*(UDOT_3+(1.2D0*UDOT_3))))
ACC4_3= (UDOT_10-(1.2D0*UDOT_1))
UDOT_6=((AUP6-(ACC4_1*vim6_1_1)+(UDOT_3+(ALPH4_2*vim6_2_2)))
ALPH5_2= ((ALPH4_2*C7)+(S7*(UDOT_3+UDOT_6)))
ALPH5_3= ((C7*(UDOT_3+UDOT_6)-(ALPH4_2*S7))
ACC5_2= ((ACC4_2*C7)+(ACC4_3*S7))
ACC5_3= ((ACC4_3*C7)-(ACC4_2*S7))
UDOT_7=((AUP7-(ALPH4_1-(0.245680927896022D0*ACC5_3)))

```

```

! USED 27.06 SECONDS CPU TIME.
! 28805 BYTES OF EXPRESSION STORAGE.
! EQUATIONS CONTAIN 514 ADDS/SUBTRACTS/NEGATES
!                               733 MULTIPLIES
!                               26 DIVIDES
!                               577 ASSIGNMENTS

```

END REGION sd_dynamics

DYNAMIC sd_states

```

Q_1' = QDOT_1
Q_2' = QDOT_2
Q_3' = QDOT_3
Q_4' = QDOT_4
Q_5' = QDOT_5
Q_6' = QDOT_6
Q_7' = QDOT_7
Q_8' = QDOT_8
Q_9' = QDOT_9
Q_10' = QDOT_10
U_1' = UDOT_1
U_2' = UDOT_2
U_3' = UDOT_3
u_4' = udot_4
u_5' = udot_5
U_6' = UDOT_6
U_7' = UDOT_7
U_8' = UDOT_8

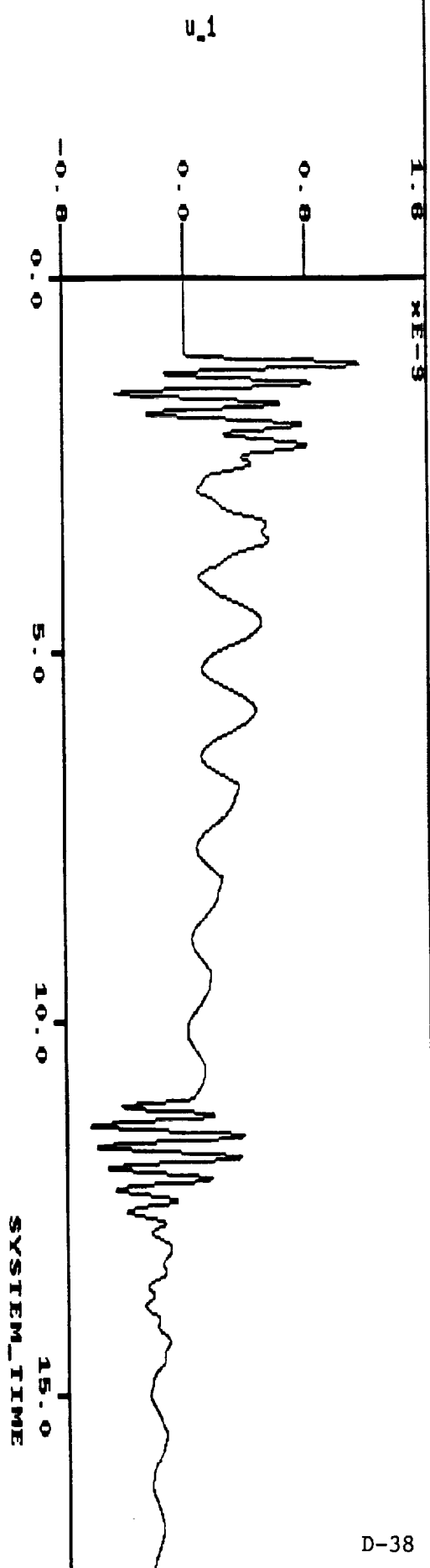
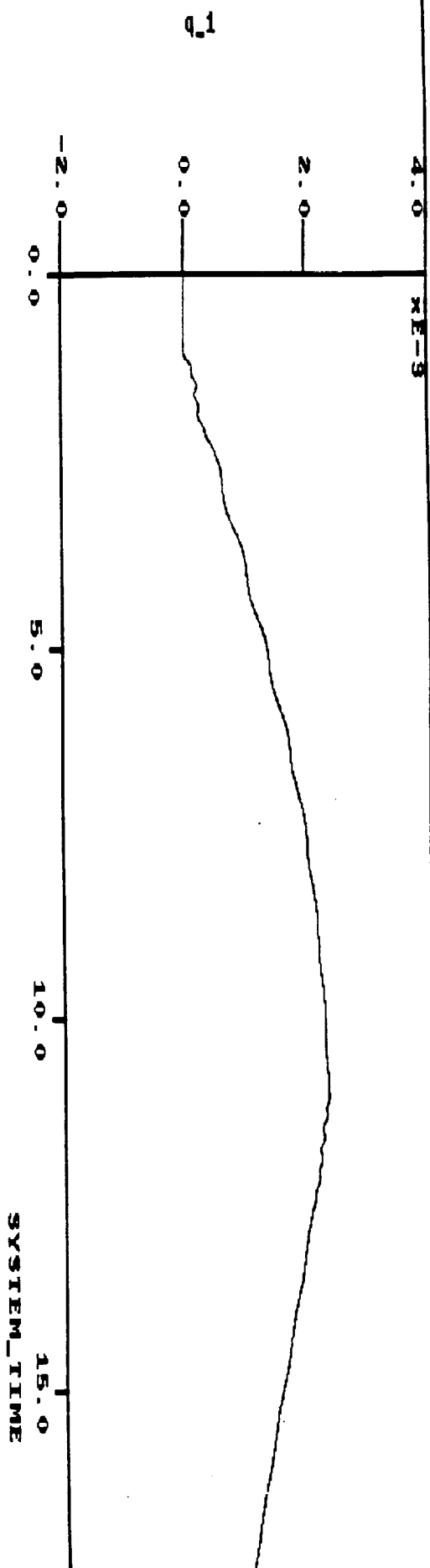
```

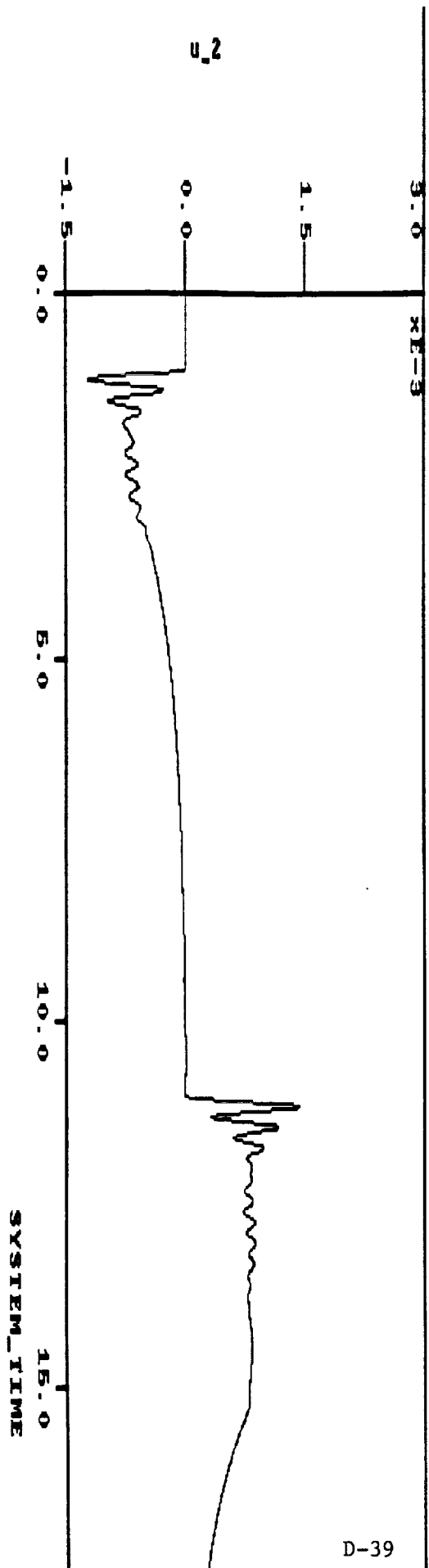
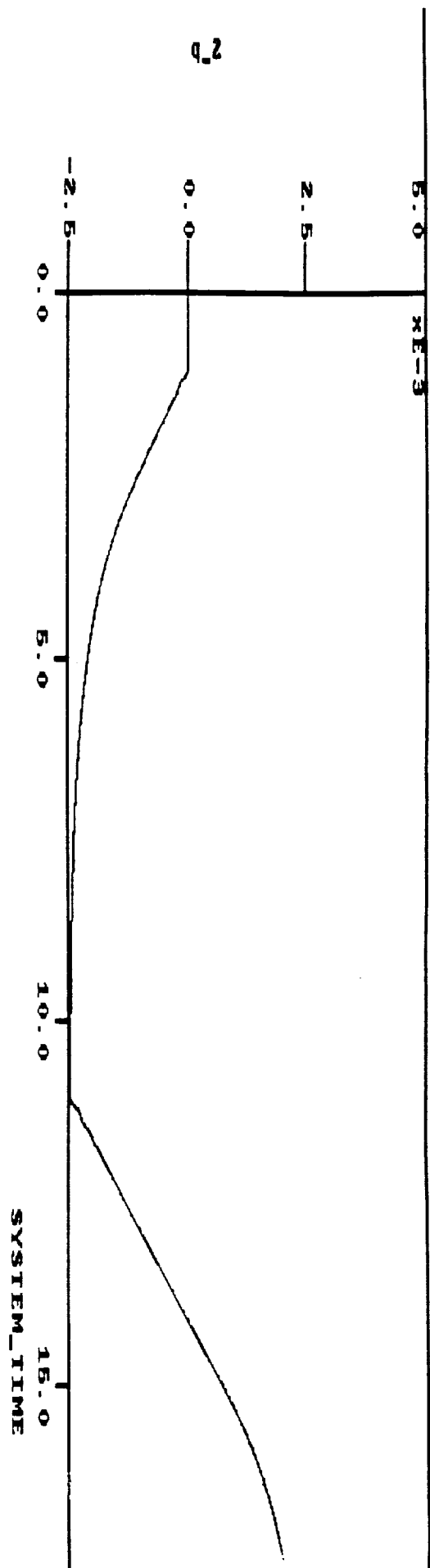
ORIGINAL PAGE IS
OF POOR QUALITY

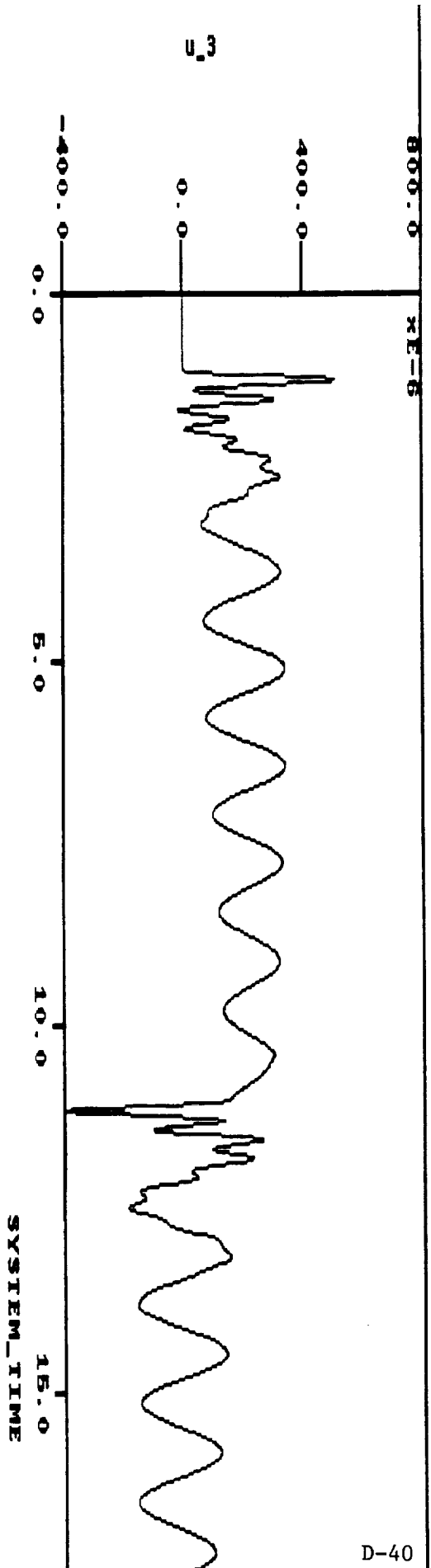
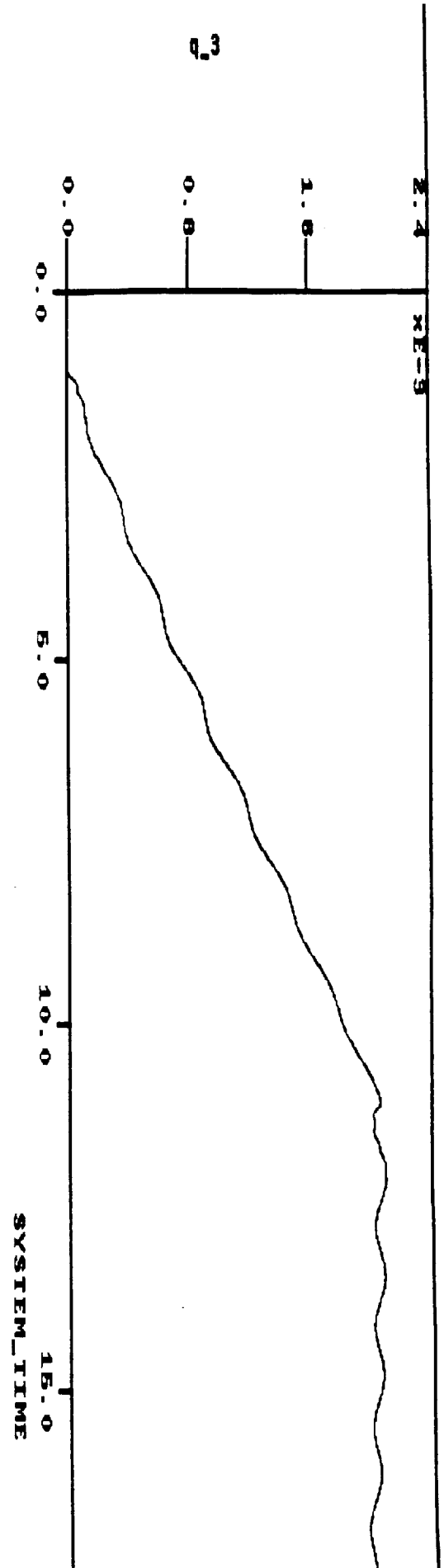
```
U 9' = UDOT 9  
U 10' = UDOT 10
```

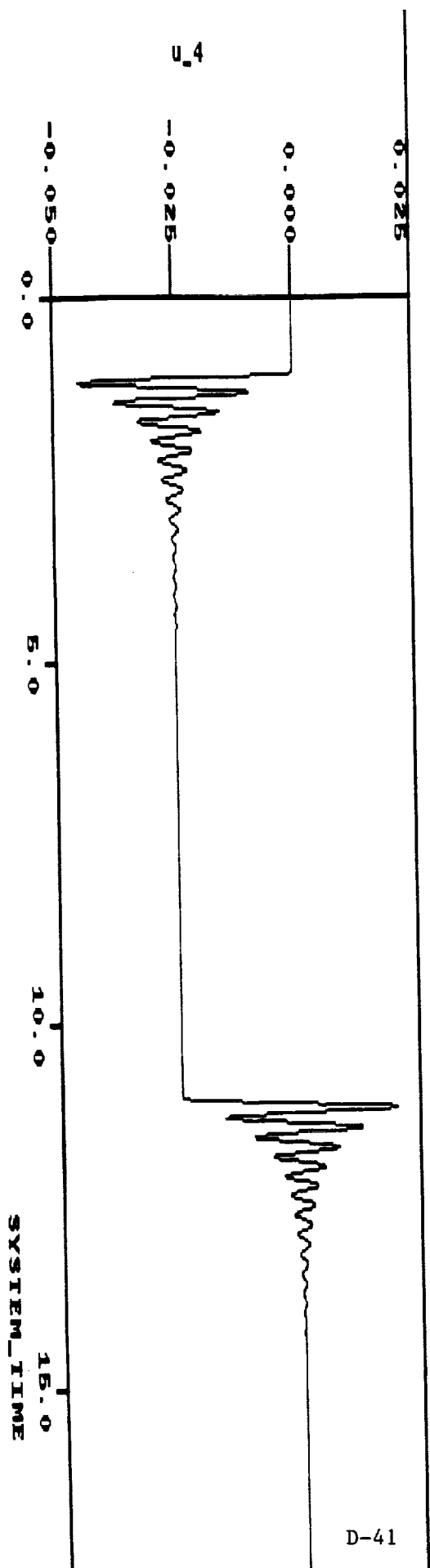
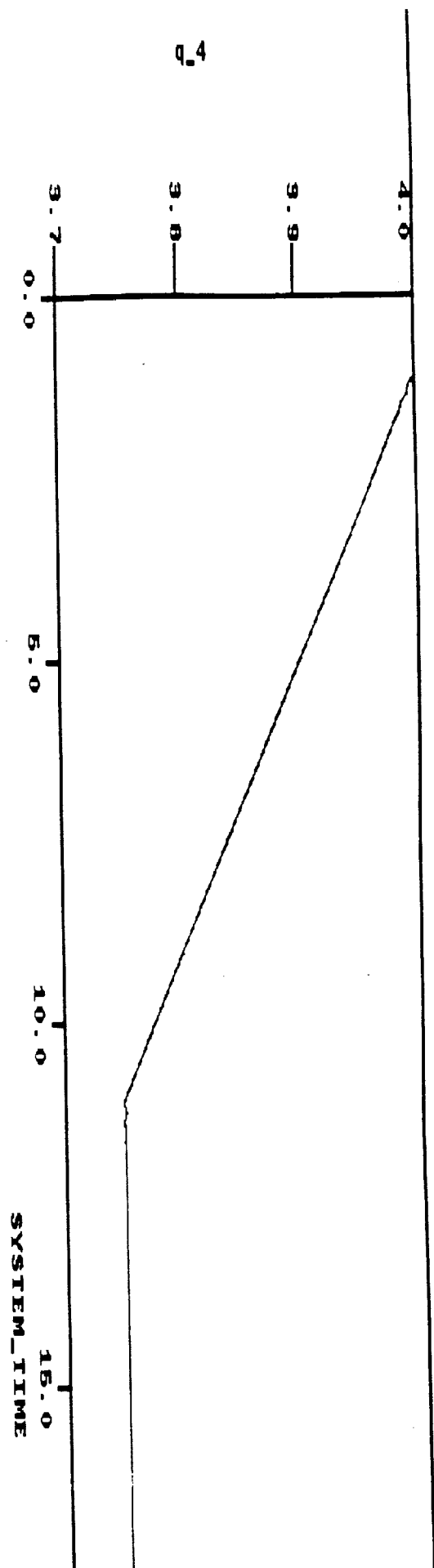
```
end_run = (system time.gt.30)
```

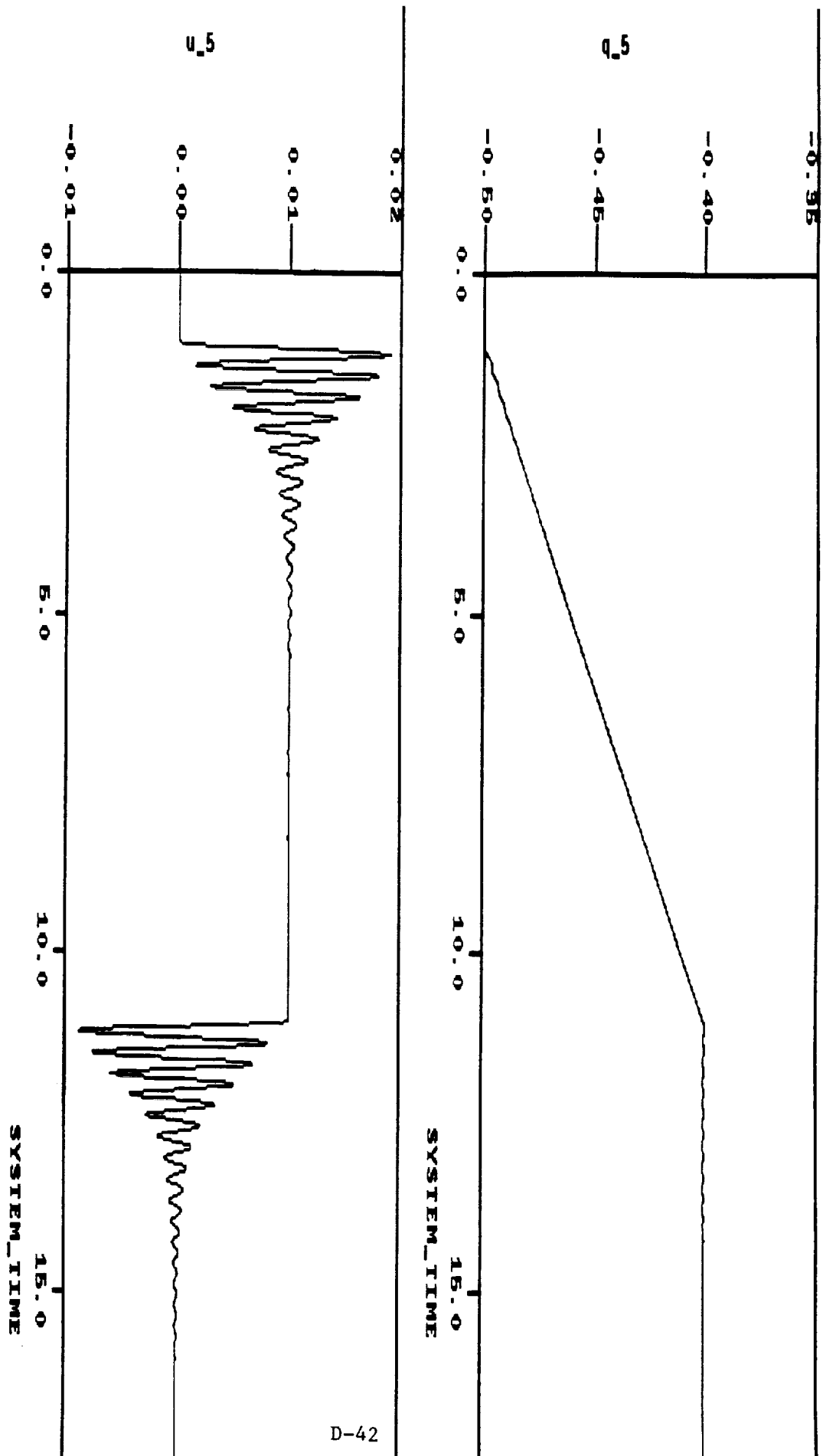
```
END DYNAMIC sd states
```

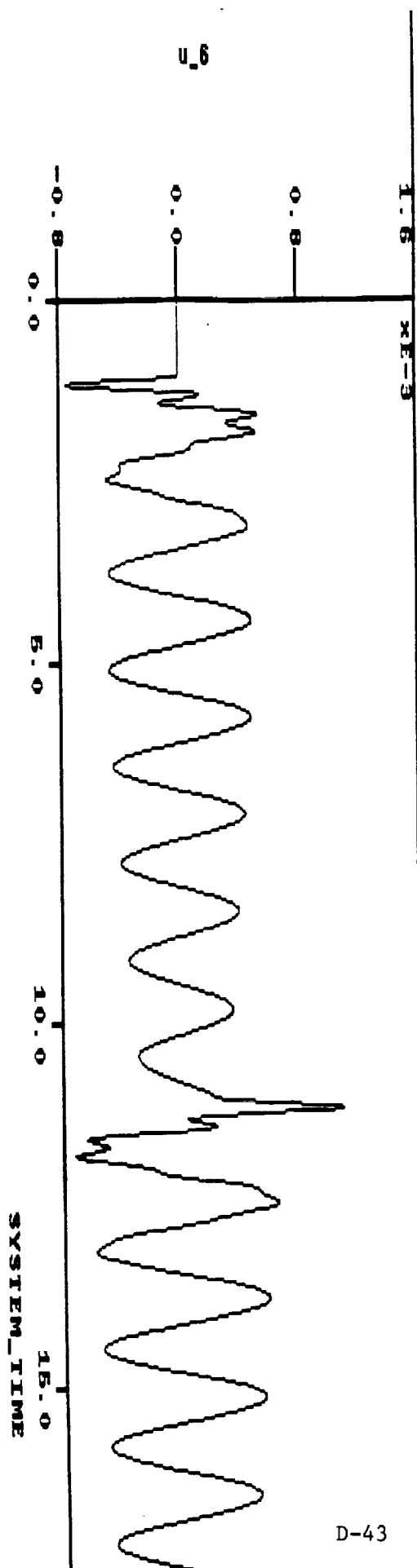
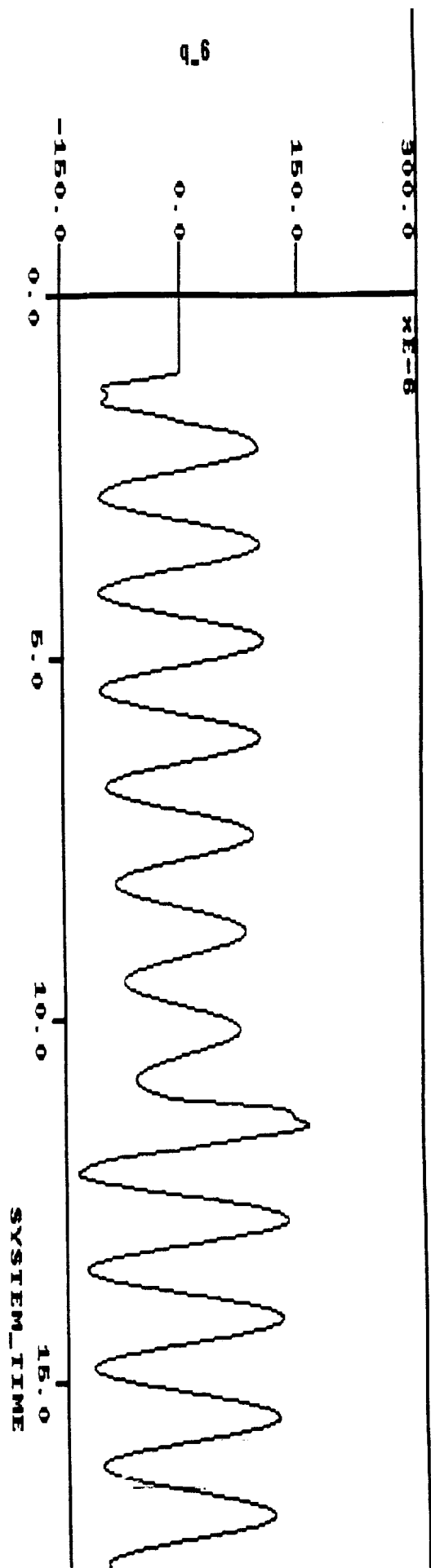


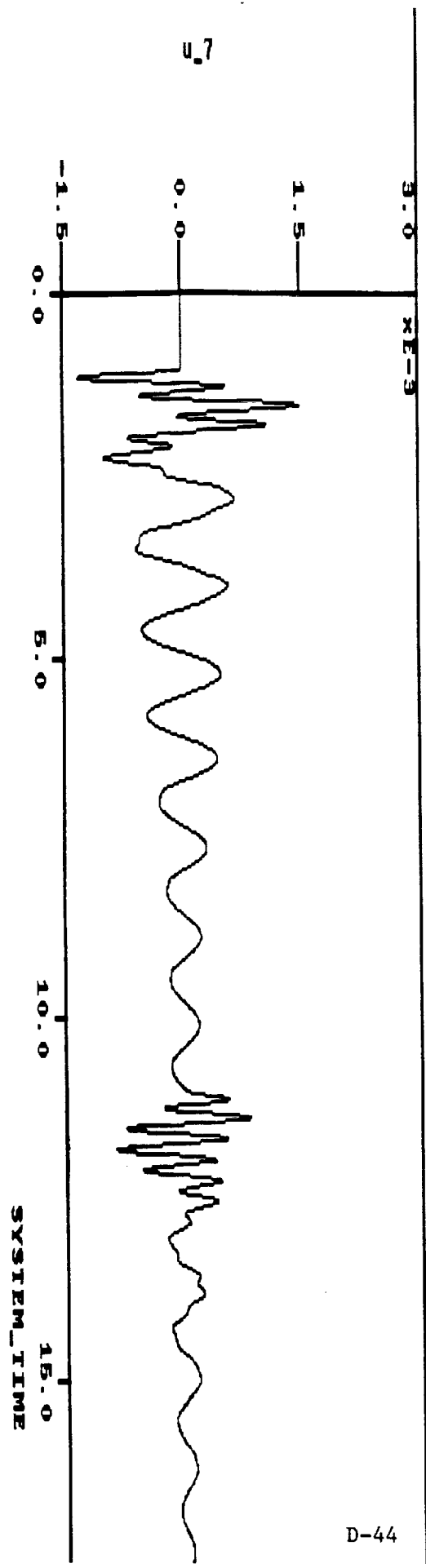
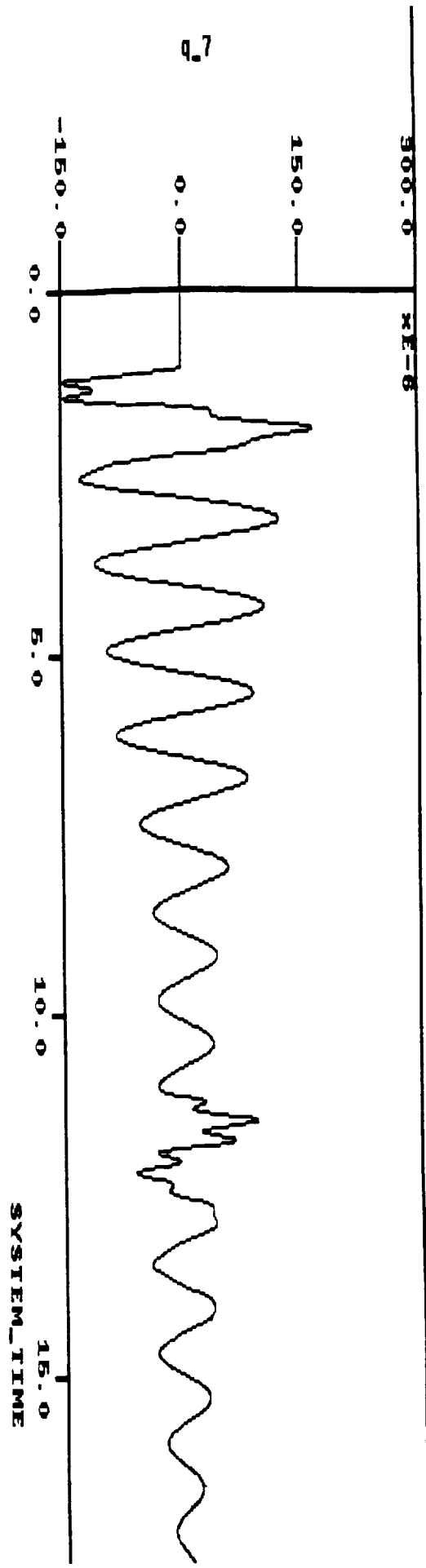












Appendix E

DADS
a
Dynamic Analysis and Design System
by
CADSI
(Computer Aided Design Software, Inc.)

DADS: Dynamic Analysis and Design System

PRODUCT DESCRIPTION

DADS is a mechanical computer-aided engineering (MCAE) software package that performs nonlinear large-displacement transient analysis and simulation. DADS is a cost-effective tool that allows you to model real-world behavior of complex mechanisms and systems *before* you build expensive prototypes.

Unique features of DADS include:

■ **Feedback, control and hydraulics:** DADS has a library of control and hydraulic components that enable you to model dynamics of feedback, control and hydraulic subsystems, their coupling with a mechanical system, and forces and torques that are fed back and act on components in the mechanical system.

■ **Flexible bodies:** DADS combines flexible body analysis capabilities with traditional rigid body dynamics, using natural modes of vibration and static correction modes that are associated with kinematic constraint reaction forces. It includes an interface processor for use with finite element structural analysis codes to calculate stiffness, mass and deformation properties.

Applications of DADS software are used in many industries, including the following: automotive, aerospace, aircraft, heavy equipment, materials handling equipment, military vehicles and robotics.

PROGRAM MODULES

DADS consists of several modules that interact to perform an analysis. Commonly used modules include:

■ **PREPROCESSOR** - The Preprocessor allows you to define a model for analysis. A command-driven menu facilitates the input and editing of model data, while data entry errors are automatically screened. Extensive on-line help and documentation for beginners and an "expert" model of data entry make the program easy to use.

■ **ANALYZER** - The Analyzer is the "heart" of DADS. A model is analyzed based on initial conditions and driving functions. Once a mechanism has been defined from the library of 2D or 3D model elements, the data set is processed and the mechanism is mathematically assembled and analyzed.

DADS solves for joint position, displacement, velocity, acceleration, total and potential energy, and internal reaction forces of models. It performs

assembly, static, kinematic, inverse dynamic and dynamic analysis.

□ **Assembly analysis** - determines whether all parts of a model can be successfully connected by the defined joints.

□ **Static analysis** - calculates the static equilibrium position and potential energy of the modeled system and components.

□ **Kinematic analysis** - calculates the relative motions of bodies in a mechanism without regard for the mass effects of the components or any forces in the system.

□ **Inverse dynamic analysis** - calculates the forces necessary to move a model through a predetermined path or given motion.

□ **Dynamic analysis** - calculates the relative motion of bodies in a mechanism, including mass effects of the components and any forces in the system.

■ **POSTPROCESSOR** - The Postprocessor presents the results of an analysis in tabular or graphical form. Plots of any variable as a function of time, body position, joint forces, acceleration, velocity, total potential energy, and all internal reaction forces can be quickly examined in various formats.

■ **DADS GRAPHIC ENVIRONMENT** - This module allows animation of analysis results. With simple assembly geometry commands, DADS Graphic Environment automatically links model geometry with position data from the Postprocessor and creates a realistic animation of a simulation. These animations allow you to quickly visualize the performance of a model, and provide an excellent tool for demonstrations and presentations. Options include wireframe and solid-shaded geometry with light sources and shadowing.

INTERFACES

DADS integrates with other MCAE applications such as FEA/FEM and CAD/CAE as part of a total design system. Interfaces are available for:

■ AutoCAD ■ PDA/PATRAN ■ MSC/NASTRAN ■ COSMIC/NASTRAN ■ SDRC/I-DEAS ■ ANSYS.

Other interfaces and custom interfaces are also available.

HARDWARE REQUIREMENTS

DADS is available on all popular engineering workstations, mainframes, super minicomputers and PCs.

STANDARD LIBRARY OF MODELING ELEMENTS

DADS contains a standard library of modeling elements from which simulation models are constructed.

Joints

■ Revolute ■ Translational ■ Cylindrical ■ Spherical ■ Universal ■ Screw ■ Revolute-Revolute ■ Revolute-Translational ■ Revolute-Spherical ■ Revolute-Cylindrical ■ Spherical-Spherical ■ Bracket ■ Planar ■ Gear

Forces

■ Tire ■ Friction ■ Bushing ■ Leaf Spring ■ TSDA (Translational Spring-Damper-Actuator) ■ RSDA (Rotational Spring-Damper-Actuator) ■ User-Defined Force

Constraints

■ Point ■ Position ■ Difference ■ Distance ■ Angle

Controls

■ Amplifier ■ Multiplier ■ Output ■ Dead Zone ■ Parameter ■ Delay ■ Sampled Delay ■ Sampled First Order ■ First Order ■ Sampled Second Order ■ Function ■ Second Order ■ General ■ Hysteresis ■ Summer ■ Input ■ Switch ■ Integrator ■ Limit ■ User-Defined Algebraic

Hydraulics

■ Accumulator ■ Hydraulic Motor ■ Check Valve ■ Double Actuator ■ Flow/Volume Update ■ Servo Valve ■ Single Actuator ■ Valve

Others

■ Driver ■ Point of Interest ■ Initial Condition ■ Rigid Body ■ Curve Data ■ Header Text ■ Flexible Body ■ Impact ■ Reference Frame ■ Road ■ Track ■ Road Wheel



P.O. Box 203 □ Oakdale, IA 52319 □ (319) 337-8968

Rev. 10/89 5M

Appendix F

Information On The
University of Iowa
Center for
Simulation and Design Optimization
of
Mechanical Systems

Members of
**Center for Simulation and Design
Optimization of Mechanical Systems**
The University of Iowa
Iowa City, IA

ALLIANT COMPUTER SYSTEMS
APOLLO COMPUTER, INC.
BOEING AEROSPACE
CADSI
CATERPILLAR, INC.
CONTRAVES GOERZ CORPORATION
U.S. NAVY DAVID TAYLOR RESEARCH CENTER
EASTMAN KODAK
EVANS & SUTHERLAND
FMC CORPORATION
FORD MOTOR COMPANY
GENERAL DYNAMICS LAND SYSTEMS
GENERAL DYNAMICS FORT WORTH DIVISION
GENERAL DYNAMICS ELECTRIC BOAT DIVISION
GENERAL MOTORS CORP. CHEVROLET-PONTIAC-CANADA GROUP
INTERGRAPH CORPORATION
J.I. CASE COMPANY
NASA GODDARD SPACE FLIGHT CENTER
NASA LANGLEY RESEARCH CENTER
PM TRADE
SILICON GRAPHICS
STELLAR COMPUTER, INC.
U.S. AIR FORCE - ROBINS AIR FORCE BASE
U.S. ARMY ARMAMENT R & D CENTER
U.S. ARMY TANK-AUTOMOTIVE COMMAND
U.S. ARMY MATERIALS TECHNOLOGY LAB
U.S. ARMY MISSILE COMMAND

NEWSLETTER

Published bimonthly for distribution exclusively to I/UCRC participants

The University of Iowa

Iowa City, Iowa 52242

Center for SIMULATION AND DESIGN OPTIMIZATION of Mechanical Systems

an NSF/ARMY/NASA-sponsored Industry/University Cooperative Research Center (I/UCRC)

THREE-YEAR PROGRESS REPORT 1987-1990

— Contents —

Center at a Crossroads	1	Center Organization	8
Progress in Meeting Center Goals	2	Current Abstracts	10
Lessons Learned	7	Calendar	12
Vision for the Future	7		

CENTER AT A CROSSROADS

After three years of operation, the Center for Simulation and Design Optimization has met many of its original three-year program goals, some with difficulty and others with a degree of success that has exceeded expectations. We have learned a number of lessons and have gained insights to guide the program in profitable directions for the future. Based on program restructuring agreed to at the Center Semi-Annual Program Review in April 1990 and organizational restructuring that has occurred during the summer of 1990, this newsletter (1) outlines Center progress in meeting original three-year goals, in terms of both software deliverables and basic technologies developed, (2) summarizes key lessons learned, (3) outlines a vision for the future of the Center, and (4) presents an organizational restructuring that dedicates management attention and resources to each of the major goals and provides a framework for efficient operation of the Center.

Three-Year Progress: As indicated in the section that follows, most of the original three-year goals of the Center have been achieved. Spectacular advances have been made in real-time dynamic simulation, interactive dynamic simulation, operator-in-the-loop simulation, and continuum-based structural design sensitivity analysis. Significant progress has been made in computer networking, visualization, dynamic stress and life prediction, dynamic system modeling, dynamic system linearization, numerical integration of differential-algebraic equations

(DAE), robotic system simulation, and track vehicle design. Software integration for multidisciplinary simulation-based design has proven to be a more substantial challenge than had been anticipated, particularly as regards communication of geometry data across different software platforms and creation of robust runstreams for multidisciplinary applications. While theoretical developments in singular configuration and mechanism workspace analysis have been developed and published, creation of interactive tools for automated workspace analysis remains a difficult and illusive goal. Finally, extension of control and hydraulics capabilities in dynamic simulation proved to be overly ambitious undertakings, given the level of effort available in the Center program.

Software deliverables, which are summarized in the following section of the newsletter, are significant as individual capabilities. To realize their full potential, however, they remain to be integrated into a broadly usable industrial environment. For example, the new general-purpose dynamics simulation code, the dynamics workstation, visualization of dynamic system software, and the network computing system are now available individually. They now need to be integrated into a qualitatively new methodology for dynamic system simulation that can serve as the foundation for a next generation product to serve a broad cross section of US industry.

Lessons Learned: The basic Center research program tenet that ~~balanced attention should be placed on advancing the underlying theory and systematically implementing~~

methods in the form of usable software has proven to be productive. It has been found that, even for experimental research software implementation, careful design and planning is essential to accomplish the intended goals. Object-oriented software design methods and programming languages that were adopted midway in the first three-year period of center operation have proved to be valuable constructs in numerous areas, but are not a panacea for resolving all challenges. In particular, trade-offs involving the supportability and generality of object-oriented design and the computational efficiency of number crunching applications are yet to be resolved.

An unanticipated finding has been that software portability across hardware platforms has become less difficult than anticipated, provided that a Unix environment is used. In contrast, portability of applications that must interface with geometric modelers has been found to be far more difficult than anticipated. The lack of standards in geometric modeling makes it very difficult to communicate geometric information among applications programs. Finally, experience shows that software implementation of methods developed in center research must be viewed in substantially different contexts and implemented at various levels of refinement and polish. The hierarchy of CAE software developed by the Center may be viewed as follows: (1) end-user-oriented software packages such as the dynamics workstation and design sensitivity analysis workstation; (2) building block software such as visualization of dynamic systems, the new general-purpose dynamic simulation code, and network computing tools; and (3) software frameworks such as the dynamic stress and life prediction runstream that must be implemented with different combinations of CAE tools in each industrial application environment. These and related lessons learned outlined later in the newsletter have guided center management in creating a practical vision for the future of the Center and in organizing the research program toward constructive ends.

Vision for the Future: The above-noted progress and lessons learned that are outlined later in this newsletter, taken together with the enormous success in operator-in-the-loop real-time simulation and the major new Iowa Driving Simulator facility that has been obtained by the Center, suggest that a refined vision for the future of the Center be established as the basis for technical planning and management. Two major opportunities, which are complementary but quite different in technical content, appear on the horizon for development by the Center. First, the emerging field of concurrent engineering, using the simulation-based design approach that has been suggested by DoD industry study groups and our Center participants, represents an area in which the Center can play a leadership role through computer-aided engineering tool development and integration to support a broad cross section of US industry. The Center's focus on dynamics, structures, and controls represents a manageable, but adequately broad class of applications to form the foundation for technology development and transfer to the practice of the profession in US industry.

The second major area of opportunity for the Center concerns operator-in-the-loop simulation, with a focus on vehicle driving simulation. The Iowa Driving Simulator

(IDS) that is being constructed by the University, using real-time simulation technology developed by the Center, will be the finest driving simulator in the US for approximately five years, until the National Advanced Driving Simulator (NADS) is constructed. To exploit this major new facility and create a unique national program in human factors research and mechanical system design for the human operator, which is at the heart of concurrent engineering, operator-in-the-loop real-time simulation is identified as the second major thrust of the Center's program.

Organizational Restructuring: While there is some synergism between the concurrent engineering and operator-in-the-loop simulation goals, guidance from Center participants through the Center Advisory Board over the past year suggests that it is important to separate the function of the IDS and ultimately the NADS from the basic Center program in concurrent engineering. This has led us to restructure the management of the Center, as indicated later in the newsletter, to provide strong focus on the basic thrusts and to better manage resources available to the Center. Professor Kyung Choi has accepted the responsibility of Associate Director for Concurrent Engineering, Professor Jon Kuhl will serve as Associate Director for Computing, and Professor James Stoner will serve as Associate Director for the Iowa Driving Simulator. This restructuring has been carefully planned during the summer of 1990, with attention to mission and function state personnel. All concerned with management of the Center are confident that the new structure will better serve the needs of participants and enhance productivity of the Center's program.

PROGRESS IN MEETING CENTER GOALS

MULTIBODY DYNAMICS

Software Deliverables

New General-purpose Dynamic simulation Code(NGDC): The β -version of the NGDC is available now and has been released to several participants, with user's and examples manuals and an installation guide. The NGDC provides open- and closed-loop rigid body analysis capabilities. Five standard joint elements and two force elements with linear and nonlinear spring, damper, and actuator forces are provided. Three different types of function generators (harmonic, polynomial, and spline curve) are available to describe nonlinear force elements. A restarting capability is provided for continuous simulation. The NGDC provides engineers the time history of position, velocity, and acceleration of each joint, in relative coordinates, and each body, in generalized coordinates. Since the code is implemented using the C++ language, the AT&T C++ compiler and a Unix operating system are required to install the NGDC. Version 1.0 of the NGDC will be available by the end of October 1990. This version of the NGDC will provide driver elements and reaction force computation capability. Complete software documents, including SPS (Software Performance Specification) and SRS (Software Requirement Specification), will be available.

Dynamics Workstation (DWS): Version 1.0 of the DWS will be available by the end of October 1990. The DWS is a modeling tool that assists engineers in defining mechanical systems in a graphics-oriented, interactive way. Many useful modeling capabilities are included, such as topological layout, automatic assembly and loop closure, joint exercising, redundant joint checking, and initial condition definition. Geometric data that are needed to run VDS and modeling data that are needed to run the recursive dynamics code and DADS have been produced. Using the DWS, journeyman engineers are able to easily create mechanical system models and evaluate their performance, using advanced simulation and animation tools. Four technical reports and one software document have been written for the DWS.

Basic Technologies

Recursive Dynamics: The recursive formulation without acceleration elimination (Order N^2) has been employed to develop the NGDC. An object-oriented software design has been developed to construct a software structure that provides for extensibility. After extensive review of the β -version code, an inter-object data access method has been created to improve efficiency. A tree-traversing method has been developed for closed-loop analysis and for system mass and force vector computation. For numerical solution, alternate types of linear equation solvers will be employed to match the characteristics of matrices used in analysis. Recursive dynamics formulations, both with and without joint relative coordinate acceleration elimination, have been developed for rigid and flexible multibody dynamics. Extensive analysis and testing of alternate formulations have provided a thorough understanding of trade-offs among formulations, for generality and computational efficiency. Extensions in conceptual formulation of flexible multibody dynamics that define system topology in terms of frames and transformations, rather than bodies and joints, have demonstrated potential for more than one order of magnitude speed-up over conventional formulations for flexible system dynamics, to support broad classes of simulation and life prediction. These developments form the foundation for the next decade of multibody dynamic simulation, in both concurrent engineering and real-time operator-in-the-loop applications.

Differential-Algebraic Equation (DAE) Solvers: Fundamental developments in methods for implicit numerical integration of DAEs have been completed and implemented, with good computational efficiency and excellent numerical stability. Generalized coordinate partitioning and tangent space parameterizations of constraint manifolds have been shown to be practical and computationally effective. Theoretical developments in error analysis have shown that the algorithms are rigorous, stable, and computationally effective. Algorithms based on these developments are currently being implemented in the Cartesian coordinate formulation by CADSI for the DADS code and in numerous applications within the Center research program.

Workspace Analysis: Theoretical foundations for mechanism singular configuration and workspace analysis have been developed and demonstrated. Singular configu-

ration manifold mapping methods developed by the University of Pittsburgh have been adapted and successfully used in Center research. While the feasibility of singular configuration and workspace analysis has been established, resources have not been available to implement general-purpose software capability for industrial application. Continued thesis research is pursuing formulation that are candidates for implementation with the new generation dynamics code.

Validation Library: To carry out validation for current flexible multibody simulation tools such as DADS, DISCOS, Order N DISCOS, and TREETOPS, a verification library system is being developed to store simulation models in a standard format, translate the standard format of the simulation model description into various simulation inputs, launch various simulation tools, and manage simulation and experimental data. A standard input data definition has been adopted. A formulation for translating standard input data into input data for DADS, DISCOS, Order N DISCOS, and TREETOPS has also been developed. Topology analyses of the mechanical system, recursive position and velocity analysis, and initial assembly based on constraint error minimization are making use of translation of the standard data. Since most flexible dynamic simulation codes are based on the deformation mode approach, generating standard data for a flexible body requires processing finite element output data to dynamic input data.

OPERATOR-IN-THE-LOOP SIMULATION

Deliverables

Iowa Driving Simulator (IDS): With the aid of DoI and NASA participants in the Center, major graphics and motion base assets have been obtained to construct the IDS. The University has provided \$1.5 million to support completion of graphics equipment and construction of a building addition to house the IDS, and integration of the system. The IDS will represent the finest driving simulator in the US and will serve a broad range of Center participants and other industrial and government organizations. An extensive technical report that defines IDS capabilities and a business plan for sustained operation have been completed and distributed to participants and other potential users of this major facility.

Basic Technologies

Real-Time Dynamic Simulation: The feasibility of real-time dynamic simulation, through parallel processing on the Alliant FX/8, was demonstrated during the first year of Center operation. Specialized parallel processing tools were developed to facilitate the development of parallel dynamics applications. These tools can organize and manage parallel execution threads with very high efficiency. The parallel processing tools expedite development of parallel applications and provide easy portability among various parallel computing platforms. These tools are currently being used in development of the new generation parallel dynamics codes.

During the second and third years of Center operation, the real-time dynamics focus shifted toward achieving real-

time dynamics performance on the newly emerging class of high-performance, multiprocessor workstations, based on reduced instruction set complexity (RISC) technology. Hewlett-Packard donated an HP/Apollo DN10000 system with four processors to the Center. The parallel processing tools developed earlier for the Alliant FX/8 have been ported to this platform, along with parallel dynamics applications that employ the tools. In the summer of 1990, dynamic simulation of a wheeled vehicle (the HMMWV) was achieved with a frame rate of 3.3 msec per integration timestep, twice the speed achieved with the eight-processor FX/8. This clearly illustrates the feasibility of real-time or faster dynamic simulation on modest-cost RISC-based multiprocessor platforms.

The Center continues to track developments in computing systems, particularly RISC-based workstations and parallel computers, with respect to their suitability for real-time dynamics. The current suite of parallel processing tools will be ported to new platforms as deemed appropriate. Parallel versions of new generation dynamics codes will be developed utilizing these tools. In conjunction with the Networking, Visualization, and Parallel Computation and Multibody Dynamics project areas, an integrated design environment, coupling high-speed parallel dynamics with on-line visualization and other forms of analysis capability, will be developed. The end result of this joint project will be an interactive simulation workstation that fully integrates modeling, real-time simulation, and quantitative analysis.

DYNAMIC STRESS AND LIFE PREDICTION

Software Deliverables

Life Prediction Software: The α -version of the life prediction software was distributed in October 1989, only for VAX/VMS operating systems. In order to increase the range of application of the life prediction software, it was ported to the Alliant with a Unix operating system. The β -version of life prediction software is under development, using the Integrated Simulation Modeling (ISM) software system as an integrating environment for wrapping finite element analysis codes, flexible multibody dynamic analysis codes, and fatigue life prediction codes. A new dynamic stress computation method will soon be implemented in this environment. The new dynamic stress computation method reflects an efficient way to calculate dynamic stresses that are induced by inertia loading.

Basic Technologies

Dynamic Stress Computation: A computational life prediction methodology for mechanical systems was initially developed using a modal stress superposition method. An integrated life prediction methodology was demonstrated. The accuracy of predicted mechanical component life was verified by experimental fatigue tests, using automotive suspension components of an off-road vehicle. The accuracy of dynamic stress computation has recently been improved, by calculating stresses from inertia load effects through the entire body. The resulting hybrid method uses flexible body dynamic analysis and a quasi-static stress computation method. The accuracy of the hybrid method for calculating dynamic stresses has been

demonstrated using numerical examples.

CONTROLS AND TELE-OPERATION

Basic Technologies

Merger of Linear Control and Multibody Dynamics: The merger of linear control and nonlinear multibody dynamics has been successfully tested using Martin Marietta's FTS robot. The control model was developed in EASY5 by Martin Marietta's FTS Controls group. Dynamics modules for the robot developed in the Center have been introduced as FORTRAN components into the EASY5 environment. This capability will be expanded to more general mechanisms that may include kinematic loop closure. This capability will expedite closing the gap between linear control design and nonlinear multibody dynamic simulation.

Linearization: The Center's linearization capability was developed in DADS Revision 4. During the past summer, it has been made compatible with Revision 6 by CADSI. For the recursive formulation, a new linearization method has been developed. Its implementation is being planned.

Tele-Operation: Recent upgrades of the Iris graphics workstation have sped tele-robotics simulation so that the RRC robot simulation runs in real-time. This is a key achievement in tele-robotic simulation, because the simulation includes dynamics, control, and graphics animation. The tele-operation simulator that was developed for the RRC robot is being modified to accommodate Martin Marietta's FTS robot. For human factors research in tele-operation, our colleagues Kevin Berbaum from Radiology and Lynn Zimba from Psychology are evaluating various types of visual feedback, such as single- vs. multi-window display and fixed vs. moving viewpoint, in combination with different hand controllers. Additionally, contact dynamics for the FTS robot is being investigated using two different approaches: (1) modeling environmental contact as kinematic loop closure, using the cut-joint idea in dynamic analysis, and (2) keeping the open-loop kinematic structure and implementing impedance control.

DESIGN SENSITIVITY ANALYSIS AND OPTIMIZATION

Software Deliverables

Design Sensitivity Analysis and Optimization Workstation (DSOW): Version 1.0 of the DSOW will be available by the end of October 1990. The DSOW provides the designer with capabilities for parameterizing line and surface design components; defining displacement, frequency, buckling, and volume performance measures for which sensitivities are to be computed; executing application software at remote machines; automating design sensitivity analysis (DSA) computation; distributing computation over a computer network; visualizing sensitivity information; automatically updating structural dimensions for a perturbed design; and integrating geometric modeling and finite element analysis codes. A user interface for the workstation has been designed and implemented with an engineering spreadsheet, using OSF-MOTIF. A refined

database system has been developed to support the design process. The user interface and computational modules of the DSOW that support the user interface have been implemented in C++. Currently, the geometric modeler that is supported by the DSOW is PATRAN. The structural analysis codes ANSYS, MSC/NASTRAN, and ABAQUS are supported. For remote process execution and distribution of DSA computation, any Unix-operating machine can be used. The portability of the prototype DSOW has been verified on an Intergraph CAD workstation. Fourteen technical reports and three software documents have been written for the DSOW.

Basic Technologies

Shape Design Sensitivity and What-if Workstation: A methodology for integrating technologies for planar and spatial structural component shape design has been developed. The feasibility of the technology has been demonstrated in a prototype Shape Design Sensitivity Analysis and What-if Analysis Workstation, which has been developed using Apollo Dialogue and FORTRAN. Capabilities developed in the Shape Design Workstation include finite element error analysis and mesh adaptation, design parameterization for curve and surface boundaries, automated boundary and domain velocity computations, automated design sensitivity computation using both direct and adjoint methods, visualization of design information, and automated geometric shape and finite element mesh updates that result from design changes.

Configuration Design Sensitivity Analysis: For built-up structures, a new DSA theory for configuration design variables has been developed, to allow layout design of built-up structures that include truss, beam, membrane, and shell design components. Once implemented, it will provide a unique capability that can support layout design of built-up structures such as vehicle, aircraft, and space structures.

DSA of Dynamic Frequency Response: A continuum design sensitivity analysis method for dynamic frequency response of structural systems has been developed. A variational approach for non-self adjoint operators, using complex variables, has been developed to obtain design sensitivities of frequency response with respect both sizing and shape design variables. This method has been tested using a large vehicle system finite element model that was developed by Ford Motor Company for NVH (noise, vibration, and harshness) design optimization. This new capability is being implemented under support from Ford Motor Company, so that it can function with a Ford proprietary NVH analysis code.

DSA of Nonlinear Structural Systems with Critical Loads: A new continuum formulation for DSA of critical loads with respect to sizing and shape design variables has been developed for nonlinear structural systems with geometric and material nonlinearities that are subjected to conservative loading. This new DSA capability can be used to obtain an optimized vehicle body design that satisfies safety requirements for vehicles under crush conditions.

DSA of Transient Structural Dynamic Response: A new continuum-based DSA theory has been developed for

transient dynamic response of built-up structures. The DSA method does not require derivatives of eigenvector and Ritz vectors. Numerical experimentation with the method indicates that it is extremely efficient and accurate.

NETWORKING, VISUALIZATION, AND PARALLEL COMPUTATION (NVPC)

Software Deliverables

Visualization of Dynamic Systems (VDS): VDS was developed by the Center during its first year of operation. Since that time, it has been extensively enhanced. VDS is currently supported on a wide range of graphics platforms. Capabilities of VDS continue to be upgraded to enhance performance and extend functionality. An IGES translator that supports a number of entities, including lines, planes, parametric surfaces, and finite element representations, is currently undergoing internal testing and will be included with the next release of VDS. Work is also continuing on a Motif-based graphical user interface for VDS and on a number of enhancements to performance and networking capabilities.

Network Computing System (NCS): The NVPC project provides the NCS - which is used by VDS - other Center deliverables to participants. The Center has ported NCS to a variety of platforms and currently can provide distribution and support for seven different platforms. The NVPC project also furnishes internal technical support for other I/UCRC project areas that use these tools, X-windows, Motif, and parallel processing tools.

Basic Technologies

Parallel Processing Tools: Specialized parallel processing tools have been developed to support the needs of high-speed dynamic simulation. Fully compatible versions of these tools are currently supported on the Center's Alliant FX/8 and HP/Apollo DN10000 multiprocessors. These tools are currently being used by the many body dynamics and operator-in-the-loop simulation projects to develop high-speed and real-time dynamic simulation codes.

Distributed Computing and User Interfaces: Center staff are actively tracking developments in the Open Systems Foundation (OSF), particularly with respect to the selection of standards for distributed computing environments and user interfaces. Tools currently in use at the Center are compatible with selections made thus far by OSF. The Center will continue to assure that computing tools used in Center projects remain compatible with prevailing OSF standards.

TOOL INTEGRATION FOR CONCURRENT ENGINEERING (TICE)

Software Deliverables

Integrated Analysis Capability (IAC): The Center has been successful in using an engineering database and module management system, called the Integrated Analysis Capability (IAC), from Boeing. IAC is used in the Center's Dynamic Stress and Life Prediction project for

engineering data management and run stream control. IAC Version 2.5 and the companion user manual are now available for delivery to participants. A VAX/VMS version and a Unix version for the Iris and Alliant FX/8 have been developed.

Basic Technologies

Integration Architecture: Technical developments to date in the TICE project have included: (1) developing design data models, (2) developing process models for each engineering analysis, (3) incorporating an object-oriented database (that stores the global model) with relational databases (that store local models for local analyses), (4) regrouping application modules into five workspaces and later extending them to eight workspaces, (5) integrating engineering tools within a concurrent engineering (CE) environment, and (6) parameterizing design products. A two-level model is used to develop a global data model for the whole CE environment and several local data models, one for each application space. Each data model defines model semantics for a specific application. Process models describe possible computational alternatives, dependencies, and flows of model characteristic data from a computational process point of view. Using process models, designers can carry out simulations in a systematic way. Simulation flow will be automated on this basis. An object-oriented database system, called ROSE (developed by RPI), will be used to store the global model that contains shared and fundamental model characteristics for CE applications. Fundamental model characteristics will be extracted for local applications. Shared model characteristics will be extracted from results of local applications and stored in ROSE for use with other applications.

Engineering Workspaces: Engineering tools are initially grouped into five workspaces: (1) a conceptual modeling workspace to define fundamental model information, such as mechanical component geometry and connectivity between components using the Dynamics Workstation and CAD systems; (2) a structural analysis workspace to analyze mechanical component characteristics from a structural point of view, using finite element modeling and analysis packages; (3) a dynamic simulation workspace to generate dynamic simulation models and scenarios and to carry out dynamic simulation and postprocessing using DWS, NGDC, DADS, and VDS; (4) a dynamic stress and life prediction workspace to calculate dynamic stresses and predict component life; and (5) a design sensitivity analysis workspace to carry out design sensitivity analysis, using sensitivity analysis and optimization portions of the DSOW. Workspaces such as a control workspace, an operator-in-the-loop workspace, and a manufacturing workspace will later be added to the integrated CE environment. Selected modules of the DICE architecture will be used with Center integration tools such as ISM and NCS. During initial development, dependence of the integrated system on DICE modules will be limited.

TRACKED VEHICLE DESIGN WORKSTATION

Software Deliverables

Tracked Vehicle Workstation (TVWS): The TVWS is under development to integrate CAD and CAE systems in

an environment where modeling tools run on workstations and computation-intensive codes run on a supercomputer. A central development in the TVWS was the creation of a database system. After a long-term study and development of a prototype, a database system that incorporates hybrid concepts of object-oriented and relational databases has been adopted and used for development. The Informax database and limited NIH classes are used to construct the database system for the TVWS.

A prototype of CAD Link, an interface to extract body and system information from the Intergraph CAD database has been developed by TACOM and used by the Center for TVWS development. Methods to reconstruct model information that is obtained from the CAD Link in an object-oriented fashion are under development. Remote database queries have been developed to allow workstations to obtain model information from a remote workstation that hosts the database. A database browser has been developed to hierarchically retrieve entities stored in the database. A DADS client is under development in TACOM for launching DADS runs on a Cray 2, from Intergraph workstations. A utility, called the Execution Monitor, that will monitor simulation status on the supercomputer during simulation is under development. Basic functionalities of fundamental utilities such as Table Editor, simple spatial animation, and x-y plotting have been developed to review and edit ASCII information and to visualize model characteristics using graphics. Basic functionalities of all of these modules will be integrated as Version 1.0 of the TVWS and delivered in December 1990.

DEPARTMENT OF TRANSPORTATION PROJECT

Software Deliverables

Vehicle Dynamics Code: A vehicle dynamics code has been written and demonstrated to be capable of analyzing both single- and multi-trailer trucks with varying axle configurations, load distributions, tire and suspension characteristics, and vehicle speeds. It accounts for tire forces and dynamic loads. The final form of the code will be validated, documented, and provided to users by November 1, 1990. The formulation of the vehicle specification is modular, to allow the analyst to assemble alternative vehicles, without direct modification of the code. A pavement model, based on a finite element code, has been developed that allows stresses and deflections to be evaluated, using predicted wheel load histories. A pavement damage and cracking model is being used to evaluate loading effects on pavement life. Model results will be compared to deflections measured in instrumented pavements at an Interstate 80 test site. The pavement and vehicle models are being implemented on a workstation with a user interface that allows the analyst to modify both vehicle and pavement layer factors and perform multiple computer runs. Input factors and performance measure outputs are generated and provided as part of the post-processor routines.

Basic Technologies

Vehicle Analysis: The vehicle analysis package utilizes high-performance planar models and a highly tuned

yet modular, code that will provide detailed time and space dynamic tire loads. The pavement models use finite element and plate models that are capable of analyzing jointed PCC concrete pavements of varying thickness and slab length.

LESSONS LEARNED

In the development and implementation of simulation and design optimization software, engineers and programmers in the Center have learned that:

- (1) Software planning is crucial for software development, even for research.
- (2) Object-oriented software design is valuable, but not a panacea.
- (3) Hardware platform portability is easier than anticipated, as long as Unix operating systems are used for all machines.
- (4) It is easier than anticipated to develop a unified interface to finite element analysis codes such as ANSYS, NASTRAN, and ABAQUS.
- (5) It is very difficult to develop a unified interface to geometric modelers, because there are no standards.
- (6) Different categories of software deliverables have very different characteristics and levels of refinement, as follows: (a) broadly usable and refined end user software, e.g., DSA workstation; (b) building block software, e.g., VDS and NGDC; and (c) software frameworks, e.g., IAC and dynamic stress and life prediction software.

To illustrate some of these points, consider the DSOW, which is a large software system that can be characterized by the following attributes:

(1) *Multi-platform Support:* As mentioned in the July newsletter, due to requests from several industry participants, the Design Sensitivity Analysis and Optimization project switched its focus in the DSOW from capability expansion to software porting across many different hardware platforms. This is complicated because the required software packages may be distributed across several machines. Currently, we are targeting Apollo, SUN, Intergraph, and DEC workstations with Unix operating systems. Some of our participants have distributed access to finite element software packages such as ANSYS, NASTRAN, and ABAQUS, thereby requiring distributed computation on machines such as Cray and Alliant mainframes with Unix operating systems.

(2) *Large and Complex Software:* The DSOW has a large database, sophisticated user interfaces, a large number of modules, sophisticated runstreams, many languages (e.g., FORTRAN, C, C++), and interfaces to many external software packages such as PATRAN, ANSYS, NASTRAN, and ABAQUS. Development of a unified interface to finite element analysis codes is easier than initially anticipated. However, development of a unified interface to geometric modelers is very difficult, because there are no standards in geometric modelers as there are in finite element analysis codes. Due to non-existence of standards in geometric modelers, it is necessary to design a different interface for each geometric modeler. Initially, the proj-

ect intended to develop the DSOW to work with PATRAN and IDEAS modelers. However, major resources are required to design and implement a new interface between DSOW and IDEAS; e.g., TACOM and DICE are supporting substantial DSA and Optimization project efforts to design and implement a new interface between the DSOW and the Intergraph/EMS modeler.

VISION FOR THE FUTURE

Technical progress in Center research and interaction with both government and industrial organizations that are pursuing significant national goals suggest that there are two complementary but quite technically different areas of future contribution from Center research.

Concurrent Engineering: First, the emerging national thrust in concurrent engineering can profit from integration of computer-aided engineering tools that are being developed and implemented by the Center. The vision for evolution of concurrent engineering presented by the DoD Industry Working Group on Concurrent Engineering of Mechanical Systems proposed that the evolution of concurrent engineering tools should first embody simulation based design, followed by evolution to a design optimization methodology. This conclusion, which was based in part on interaction of the DoD/Industry Working Group with our Center activity suggests that the Center can play a significant role in development and implementation of concurrent engineering tools for mechanical system design. The role and scope of our Center in this major development, however, must be carefully considered and designed in practical terms.

The major DARPA Initiative in Concurrent Engineering (DICE) is creating a framework for concurrent engineering of broad classes of engineering systems and is including an emphasis on manufacturing. Our Center program, in contrast, has focused on engineering tool integration to support concurrent engineering of mechanical systems. Experience gained to date in implementation of dynamic stress and life prediction software and in the TACOM pilot project on tracked vehicle design suggests that the scope of our Center activity is adequately broad to make a major contribution to the field of concurrent engineering of mechanical systems. It is also technically coherent enough to permit the Center to make significant and practical contributions to the field. Joint projects being undertaken with the DICE program will provide the Center an opportunity to contribute tools to a framework that our participants may ultimately expect to use in concurrent engineering

of their products. Developing such partnerships further permits our Center to concentrate on its strengths work within its physical and manpower limitations, and take advantage of capabilities that are being developed in large programs that are beyond the scope of our Center.

Apart from the real-time operator-in-the-loop simulation emphasis in the Center program, virtually all of the basic Center program falls well within the broad goals of concurrent engineering. With minimal project restructuring-agreed to at the April 1990 Semi-annual Program Review, the Center program can be continued and strengthened to develop and transfer concurrent engineering design

tools to our participants, in the context of the frameworks for concurrent engineering that are being developed within some of our participant organizations and by the DICE program. The organizational realignment of the Center presented in the next section of this newsletter reflects Center management's intention to maintain a major thrust in concurrent engineering tool development and integration.

Operator-in-the-Loop Simulation: Landmark developments in the area of real-time dynamic simulation of mechanical systems in the Center during its first three years of operation have provided the key enabling technology to support broad new classes of application of operator-in-the-loop simulation. Smithsonian Institution and Society of Automotive Engineers awards to the Center for this development have reinforced the unique contribution made by the Center in creating this qualitatively new engineering development and human factors research capability. The Center's efforts with the US Department of Transportation, jointly funded by the state of Iowa, have led to congressional approval of the National Advanced Driving Simulator (NADS), which will be located at a major research university in the US.

The University of Iowa is currently preparing a proposal, due in mid-January 1991, to win assignment as the NADS host institution. In addition, the University has obtained approximately \$6 million worth of advanced graphics and motion-base equipment and has allocated \$1.5 million for construction of a facility to house the Iowa Driving Simulator (IDS). The facility is now under construction and will be complete in February 1991. This major new research facility will be the best ground vehicle driving simulator in the US from its completion in 1991 until 1996, when the NADS becomes the world's best advanced driving simulator. The IDS, with minor enhancements and upgrades, will be comparable in capability to the current state-of-the-art driving simulator that is operated by Daimler-Benz in Berlin. The University of Iowa team is establishing joint relations with the Daimler-Benz research group and with a joint government/industry group in France that intends to build an advanced driving simulator in approximately four years. The University of Iowa simulator team, which has been spawned by our Center, is playing a major role in evolving the new capability for vehicle driving simulation, both in the US and elsewhere in the world.

Complementing the vehicle driving simulator application of operator-in-the-loop simulation, the Center has implemented interactive simulations of telerobotic systems with the NASA/Goddard Space Flight Center and is exploring significant joint work with a research group at The University of California-Davis in highway repair automation. With high-speed multiprocessor superworkstations in the Center laboratory and the major visualization and motion systems that are being integrated into the Iowa Driving Simulator, the Center and University have a unique opportunity to play a leadership role in a broad range of remote manipulator design and human factors research applications.

Due to the unique character of operator-in-the-loop simulation and the major resources required to implement and support the Iowa Driving Simulator and the National Ad-

vanced Driving Simulator, Center participants have agreed that management of the large-scale driving simulators should be separated from the basic Center program in concurrent engineering, for purposes of effective and dedicated management. The management restructuring presented in the next section of this newsletter has been designed to dedicate attention to both the concurrent engineering and operator-in-the-loop simulation objectives of the Center, maintaining a strong focus on Center concurrent engineering tool development goals and providing Center participants with benefits of the operator-in-the-loop simulation activity.

CENTER ORGANIZATION

To accommodate the increased scope of Center activities and to improve the management of projects, the Center management structure has been refined. The objective of the new organizational structure is to assure that:

- (1) Projects collectively implement the CCAD plan.
- (2) Inter-project working relationships are identified and maintained.
- (3) Projects are involved in proper technical issues to achieve project goals.
- (4) There is not unnecessary overlap or duplication among projects.
- (5) Common computer science support is identified and provided for multiple projects.

The Center for Computer Aided Design (CCAD) has been divided into four major programs, shown in Fig. 1: (1) the I/UCRC, (2) Special Projects, (3) Computing/Software Support, and (4) the Iowa Driving Simulator. Each program now has its own Associate Director, who manages the associated program and assists the Director of CCAD in overall management. The Associate Directors have administrative and technical responsibilities for each program. Professors Kyung Choi, Jon Kuhl, and Jim Stoner have accepted responsibilities as Associate Directors for Concurrent Engineering, Computing, and the Iowa Driving Simulator. Ed Haug will serve as Acting Director of Special Projects, until a dedicated person can be hired. Under the I/UCRC, there are eight projects: Multibody Dynamics (Ed Haug); Operator-in-the-Loop Simulation (Jim Stoner); DSA and Optimization (Kyung Choi); Tool Integration for Concurrent Engineering (Kirk Wu); Controls and Tele-Operation (Harry Yae); Networking, Visualization, and Parallel Computation (Jon Kuhl); Dynamic Stress and Life Prediction (Fook Choong); and TACOM Pilot CAE Project (Kirk Wu). Tool Integration for Concurrent Engineering and Controls and Tele-Operation are new projects in the I/UCRC that were agreed to at the April 1990 Program Review.

Individual projects that the Center carries out under additional sponsorship of I/UCRC participants or other sponsors are treated as special projects. Currently there are five such special projects: NADS, Ford NVH Design Optimization, DICE Project, JI Case Project, and ITC Suspension Design Project. To consolidate computer hardware and computer science support that are necessary for several projects, the Computing/Software Support Program has been created. In this program, there are three projects: Graphics Facility, Computer Operations, and Participant Software Support. Centralizing support re-

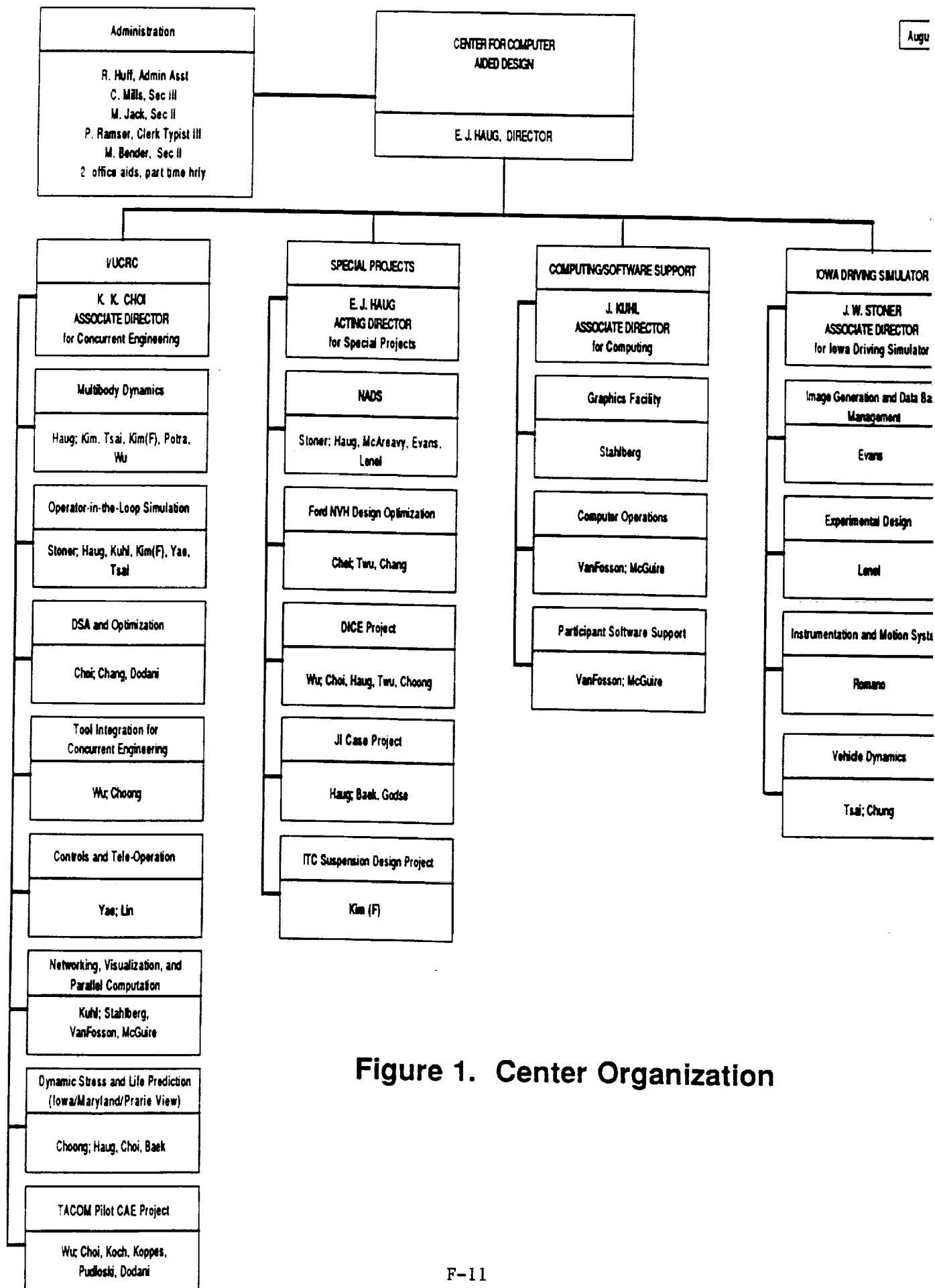


Figure 1. Center Organization

quired by several projects has resulted in more efficient use of Center resources. The Iowa Driving Simulator Program has been developed to manage and operate the recently acquired Iowa Driving Simulator (IDS). There are four projects in this program: Image Generation and Database Management, Experimental Design, Instrumentation and Motion Systems, and Vehicle Dynamics.

To strengthen the Center's organizational structure and inter-project cooperation, each project has developed Mission and Function Statements (MFS) and has defined principal working relationships with other projects. Principal working relationships identify common computer science support for each project and engineering support that the projects provide and receive to achieve their missions. The Center will next document MFS for all projects and Associate Directors. The resulting document will be used for internal program review to assure that projects accomplish their missions.

----- Current Abstracts -----

Complete Reports can be found in Volumes 5 and 6 of the Participant Technical Report Notebook

REAL-TIME DYNAMIC SIMULATION OF A ROBOT SYSTEM WITH ROTOR DYNAMICS

Li-Ping Chuang and Sung-Soo Kim
Center Technical Report R-67

Rotor dynamics are important for robots that are driven by motors and high gear ratio transmission systems. New generic recursive dynamic formulations of robotic systems with motor rotor effects are presented, where motors can be mounted on inboard or outboard bodies of the actuated joints. An organized form of the system dynamics equations is obtained through adoption of a state vector representation in equation derivation. The influence of rotors on the overall system dynamics is identified. Parallelism of the derived dynamics formulations is then exploited and a real-time parallel computational algorithm is proposed. A seven degree-of-freedom telerobot is used to illustrate rotor dynamics effects in the computational model. A computational speed that is faster than real-time simulation is demonstrated.

MANAGEMENT OF HETEROGENEOUS PARALLELISM ON SHARED MEMORY MULTIPROCESSORS

Athar B. Tayyab and Jon G. Kuhl
Center Technical Report R-68

This report considers the problem of management of heterogeneous parallelism on shared memory parallel processing systems. Heterogeneous parallelism is defined as those sources of application parallelism not associated with loops. This may include both explicitly coded and compiler generated forms. It is argued that support

mechanisms are needed to efficiently manage heterogeneous subcomputations at run-time, and that the important issues in the design and implementation of these mechanisms are different from those previously studied for support of loop level-parallelism. An empirical study of an actual application is presented. This study indicates that the specific nature of run-time support mechanisms can dramatically impact the performance of a parallel program. The study suggests that careful attention must be given to the choice of an appropriate set of run-time management mechanisms to support compiler generated or explicit heterogeneous parallelism. The use of simple, syntactically closed constructs is suggested.

A RECURSIVE APPROACH AND PARALLEL IMPLEMENTATION FOR THE ANALYSIS OF INTEGRATED CONTROLLED MECHANICAL/ROBOTIC SYSTEMS

Li-Ping Chuang and George M. Lance
Center Technical Report R-69

A general parallel computational methodology for the high-speed simulation of either integrated controlled mechanical systems or independent control systems is developed and implemented. A general control system preprocessing method is provided in order to perform the control system topological analysis and to automatically set up the parallel computational algorithm. The control system, hydraulic system and mechanical system are integrated in the analysis. A parallel computational algorithm is developed for the integral control/hydraulic/mechanical system analysis. A general-purpose simulation code is generated so that the complete control/hydraulic/mechanical system can be simulated as a single integrated model. A recursive computational method for integrated controlled robotic system analysis is also developed. New recursive formulations for robotic transformations and for the dynamics of robots with rotor effects provide a general analysis capability in robotic applications. The forward and inverse kinematics are formulated, using a recursive approach. The so-called robotic Jacobian and inverse Jacobian are also derived. Two sets of recursive dynamics equations of robots with rotor effects are formulated: the first set is for robots with rotors mounted on the outboard bodies of the acting joints, and the second set is for robots with rotors mounted on the inboard bodies of the acting joints. Equations for a typical DC motor model and a transmission model are also formulated. A trajectory planning method in Cartesian space is developed. The method generates a straight line between any two user-specified points in Cartesian space and a smooth transition trajectory between two neighboring line segments. Different position and hybrid position/force control methods are studied and tested. A resolved acceleration hybrid position/force control method, based on the recursive inverse kinematics and dynamics formulations, is designed and implemented. Joint space compliance selection matrices are derived and used in the derivation of this resolved acceleration hybrid controller. A parallel computational algorithm for the integral analysis of robotic systems under position/force control is developed and implemented. Fine grain parallelism is fully exploited in this method. The

efficiency of this algorithm is demonstrated through simulations on the Alliant FX/8 multiprocessor.

INTEGRATION OF CONTROLLED MULTIBODY
MECHANICAL SYSTEMS WITH NON-STIFF
MECHANICAL SUBSYSTEMS AND
STIFF CONTROL SUBSYSTEMS

Shih-tin Lin and George M. Lance
Center Technical Report R-70

The simulation of controlled multibody mechanical systems, with nonstiff mechanical subsystems and stiff control subsystems, usually suffer from the long run times. In this report, a hybrid integration scheme that uses separate algorithms for the mechanical and control subsystems is presented. A double pendulum driven by two AC motors and a backhoe mechanism driven by three hydraulic actuators were used to demonstrate the computational efficiency of the proposed integration scheme.

COMPUTATIONAL LIFE PREDICTION METHODOLOGY
FOR MECHANICAL SYSTEMS USING DYNAMIC SIMULATION,
FINITE ELEMENT ANALYSIS,
AND FATIGUE LIFE PREDICTION METHODS

Woon K. Baek and Ralph I. Stephens
Center Technical Report R-71

This study deals with an integrated life-prediction methodology using dynamic simulation, finite element analysis, the fatigue life-prediction method, and experimental validation for the finite life evaluation of a mechanical system. As a practical example, a multi-body dynamic model of an existing ground vehicle was developed using kinematic joints and components. The vehicle model was hypothetically run over a measured road profile at constant speed. From this dynamic analysis, load histories were obtained for each component. A lower control arm, which is a critical suspension component, was selected for the component fatigue life prediction, and a finite element model of this component was developed. Several high-stress regions were identified from the finite element stress analysis. Local notch stresses at each high-stress region were then obtained at potential fatigue crack "initiation" points, which can be called potential fatigue-critical locations. Dynamic stress histories at potential fatigue-critical locations were produced by the quasi-static approach. The local strain-life method was used to predict fatigue life of each potential fatigue-critical location. The fatigue life was defined as the typical crack "initiation" life of a crack about 2 mm in length. The fatigue life of the component was defined as the shortest fatigue life among several potential fatigue-critical locations. To validate this computerized procedure, the lower control arm was experimentally tested for stress and fatigue durability. The brittle coating method was used to identify high-stress regions. Experimental stress analysis was carried out using strain gauges.

The experimental results and predicted results based upon finite element analysis were very close: every comparison showed a difference of less than 5 percent. Also, fatigue durability tests were done for the component by repeatedly applying the same load history that was applied to the finite element model, until a 2 mm long fatigue crack formed. The breakthrough in this integration is that dynamic stress and fatigue life can be predicted in early design stages without experimental measurement of either loads or stresses. This methodology can be extended to design optimization, based upon durability.

REAL-TIME OPERATOR-IN-THE-LOOP
SIMULATION OF MULTIBODY SYSTEMS

Joe Lan Chang, Sang-Sup Kim,
and Edward J. Haug
Center Technical Report R-72

This thesis presents a general approach to achieving real time operator-in-the-loop simulation for multibody dynamic systems. Emerging real-time dynamic simulation methods are used to demonstrate the potential for creating interactive design workstations and for teleoperating space robots, with a human operator in the control loop. The recursive formulation of multibody system dynamics with relative coordinates is employed for efficient numerical analysis and implementation on a parallel computer. High-speed computer graphic techniques are employed to create realistic visual cues for the simulator. A simulator is developed in this research by integrating the real-time dynamics program, a realistic graphics display and the operator's control interface. Real-time operator-in-the-loop simulation is analyzed, as regards the goal of real clock time, not only with respect to dynamic simulation but also with respect to graphics display and the operator interface. Synchronization of the simulation is found to be most important for realism of the simulator. A backhoe simulation is implemented to demonstrate the capability for operator-in-the-loop simulation. The simulator is developed by modeling backhoe dynamics and hydraulic systems with the recursive formulation to achieve real-time simulation, developing an interactive graphics program for visual cues, and interfacing the operator's control action with the dynamic simulation through a pair of joysticks. The simulator is also implemented for teleoperated simulation of space robots.

CONTINUUM DESIGN SENSITIVITY
ANALYSIS OF STRUCTURAL DYNAMIC
RESPONSE USING RITZ SEQUENCE

Kyung K. Choi and Semyung Wang
Center Technical Report R-73

In this report, a unified continuum-based sizing Design Sensitivity Analysis (DSA) method is developed for the transient dynamic response of built-up structures by taking design derivatives of the variational equation of the built-up structure to obtain a variational equation of the

design sensitivity of the transient response. The direct differentiation method of DSA is used. The very same Finite Element Analysis (FEA) model that is used to obtain an approximate solution of the variational equation of the built-up structure is used to obtain an approximate solution of the variational equation of the design sensitivity. For large size built-up structures, the same superposition method that is used to reduce the dimension of the matrix equation of the built-up structure is used to reduce the dimension of the matrix equation of the design sensitivity. For accuracy of analysis results and efficiency, combinations of eigenvectors and Ritz vectors are used as bases. The continuum-based design sensitivity analysis method can be implemented outside established FEA codes using postprocessing data only, since the method does not require derivatives of the stiffness, damping, and mass matrices. Moreover the method is efficient since it does not require derivatives of basis vectors. Two examples are presented to demonstrate accuracy of the method. Examples treated in this paper indicate that the same number of basis vectors that are used for analysis of the built-up structure is enough for analysis of the design sensitivity.

LINEARIZATION IN THE RECURSIVE DYNAMICS FORMULATION

Tsung-Chieh Lin and K. Harold Yae
Center Technical Report R-74

The non-linear equations of motion in multi-body dynamics pose a difficult problem in linear control design. It is therefore desirable to have linearization capability in conjunction with a general-purpose multibody dynamics modeling technique. A new computational method for linearization is obtained by applying a series of first-order analytical approximations to the recursive kinematic relationships. The method has proved to be computationally more efficient. It has also turned out to be more accurate because the analytical perturbation requires matrix and vector operations by circumventing numerical differentiation and other associated numerical operations that may accumulate computational error.

RECURSIVE LINEARIZATION OF MULTIBODY DYNAMICS AND APPLICATION TO CONTROL DESIGN

Tsung-Chieh Lin and K. Harold Yae
Center Technical Report R-75

The non-linear equations of motion in multi-body dynamics pose a difficult problem in linear control design. One solution is to have the non-linear model linearized. Intuitively, the simplest linear model would be an extraction from a non-linear model by omitting non-linear effects such as Coriolis force, centrifugal force, and other forces interacting between bodies. Such simplification is rather an ad hoc approach, depending on the problem at hand. It is therefore desirable to have linearization capability in conjunction with a general-purpose multibody dynamics

modeling technique. The linearization developed in this research starts with non-linear equations of motion written in the Newton-Euler form, which are easy to construct; then, they are transformed into joint coordinate representation in two steps: Cartesian variables into state variables, and state variables into joint variables. At each transformation the kinematic relations are replaced with their linearized relations. A new computational method for linearization is obtained by applying a series of first-order analytical approximations to the recursive kinematic relationships. The method has proved to be computationally more efficient. It has also turned out to be more accurate because the analytical perturbation requires matrix and vector operations by circumventing numerical differentiation and other associated numerical operations that may accumulate computational error. The power of the proposed linearization algorithm is demonstrated, in comparison to a numerical perturbation method, through four robotic manipulators. Also demonstrated is its application to control design. In addition, a parallel algorithm for the linearization is also discussed. When the algorithm is implemented on Alliant/FX8, a shared memory 8-processor machine, it takes 16.5 milliseconds to linearize the dynamic model of a 7 degree-of-freedom robot manipulator.



October 30-31

Second Annual
Symposium on
Mechanical
System Design
in a Concurrent
Engineering
Environment

November 1-2
Semi-Annual Program Review

April 11-12, 1991
Semi-Annual Program Review

Center Director
Edward J. Haug

Associate Directors
Kyung K. Choi
Jon Kuhl
James W. Stoner

Administrative
Assistant
Rozanne Huff

Secretarial Support
Cindy Mills
Mary Jack
Pam Ramser
Mary Bender

Appendix G

A Work Breakdown Structure For Implementing Recommendation

The enclosed work breakdown structure (WBS) includes all the items contained in the recommendation section of the report, where they are discussed. It is broken into two major pieces. The 3.1 piece deals with efforts that can or are being done within current MSFC programs. Thus, they would require relatively modest expenditures and required resources could be estimated based on historical data. The 3.2 piece deals with new initiatives which would require new MSFC commitments of dedicated to-the-purpose personnel and funds. Because these would be new commitments, there are no historical data on which to base their cost (although certainly someone would have to be assigned more or less full time to focus the activity and funds made available for the new initiatives).

